IBM® DB2 Universal Database™

Consulta de SQL Volumen 1

Versión 8.2

SC10-3730-01

IBM® DB2 Universal Database™

Consulta de SQL Volumen 1

Versión 8.2

SC10-3730-01

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

Esta publicación es la traducción del original inglés *IBM DB2 Universal Database, SQL Reference Volume 1, Version 8.2, (SC09-4844-01).*

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

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

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

Contenido

Acerca de este manual

El manual Consulta de SQL en sus dos volúmenes define el lenguaje SQL que utiliza DB2 Universal Database Versión 8 e incluye:

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

A quién va dirigido 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 aquellos que acceden a las bases de datos utilizando 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.

Estructura de este manual

Este manual contiene información sobre los siguientes temas principales:

- El Capítulo 1, “Conceptos”, en la página 1 explica los conceptos básicos de las bases de datos relacionales y de SQL.
- El Capítulo 2, “Elementos del lenguaje”, en la página 61 describe la sintaxis básica de SQL y los elementos del lenguaje que son comunes a muchas sentencias de SQL.
- El Capítulo 3, “Funciones”, en la página 239 contiene diagramas de sintaxis, descripciones semánticas, normas y ejemplos de utilización de las funciones de columna y funciones escalares de SQL.
- El Capítulo 4, “Consultas”, en la página 469 describe los distintos formatos de una consulta.
- El Apéndice A, “Límites de SQL”, en la página 525 muestra las restricciones de SQL.
- El Apéndice B, “SQLCA (área de comunicaciones SQL)”, en la página 533 describe la estructura SQLCA.
- El Apéndice C, “SQLDA (área de descriptores de SQL)”, en la página 539 describe la estructura SQLDA.
- El Apéndice D, “Vistas de catálogo”, en la página 549 describe las vistas de catálogo de la base de datos.
- El Apéndice E, “Sistemas federados”, en la página 671 describe las opciones y las correlaciones de tipos para sistemas federados.
- El Apéndice F, “La base de datos SAMPLE”, en la página 725 describe las tablas de muestra utilizadas en los ejemplos.
- El Apéndice G, “Nombres de esquema reservados y palabras reservadas”, en la página 741 contiene los nombres de esquemas reservados y las palabras reservadas correspondientes a los estándares SQL de IBM y SQL99 ISO/ANSI.

- El Apéndice H, “Interacción de los activadores con las restricciones”, en la página 745 explica la interacción de los activadores y las restricciones referenciales.
- El Apéndice I, “Tablas de Explain”, en la página 749 describe las tablas Explain.
- El Apéndice J, “Valores de los registros de EXPLAIN”, en la página 777 describe la interacción entre sí de los valores de registro especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT y con los mandatos PREP y BIND.
- El Apéndice K, “Tablas de excepciones”, en la página 785 contiene información sobre tablas creadas por el usuario que se utilizan con la sentencia SET INTEGRITY.
- El Apéndice L, “Sentencias de SQL que se permiten en rutinas”, en la página 791 lista las sentencias de SQL que es posible ejecutar en rutinas con diferentes contextos de acceso a datos SQL.
- El Apéndice M, “CALL invocada desde una sentencia compilada”, en la página 795 describe la sentencia CALL que puede invocarse desde una sentencia compilada.
- El Apéndice N, “Consideraciones sobre el código UNIX ampliado (EUC) en japonés y chino tradicional”, en la página 801 muestra las consideraciones a tener en cuenta cuando se utilizan los juegos de caracteres Extended UNIX Code (EUC).
- El Apéndice O, “Especificaciones de formato de Backus-Naur (BNF) para los enlaces de datos”, en la página 809 contiene las especificaciones del formato Backus-Naur (BNF) para archivos DATALINK.

Una breve visión general del volumen 2

El segundo volumen de Consulta de SQL contiene información sobre la sintaxis y la semántica de las sentencias SQL. A continuación se describen brevemente los capítulos específicos del volumen:

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

Lectura de los diagramas de sintaxis

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

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

El símbolo ▸▸── indica el inicio de un diagrama de sintaxis.

El símbolo ──▸ indica que la sintaxis continúa en la línea siguiente.

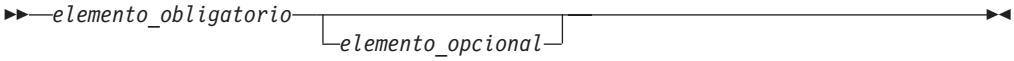
El símbolo ▸── indica que continúa la sintaxis de la línea anterior.

El símbolo ──▸◂ indica el final de un diagrama de sintaxis.

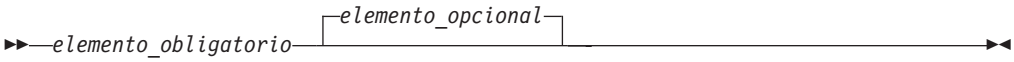
Los fragmentos de sintaxis empiezan con el símbolo ├── y terminan con el símbolo ──┤.

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

Los elementos opcionales aparecen debajo de la línea principal.

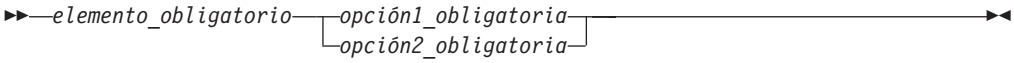


Si aparece un elemento opcional por encima de la línea principal, dicho elemento no tiene ningún efecto en la ejecución y sólo se utiliza para facilitar la lectura.

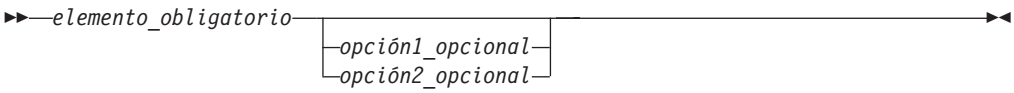


Si puede elegir entre dos o más elementos, aparecen en una pila.

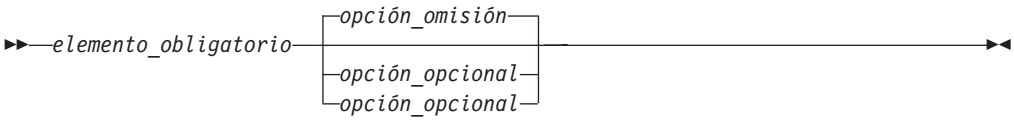
Si *debe* elegir uno de los elementos, un elemento de la pila aparece en la ruta principal.



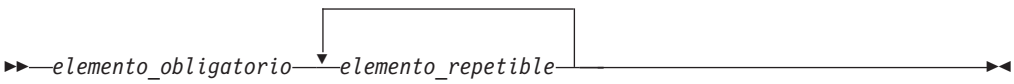
Si la elección de uno de los elementos es opcional, toda la pila aparece por debajo de la línea principal.



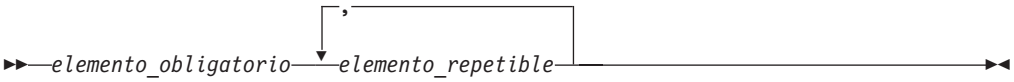
Si uno de los elementos es el valor por omisión, aparecerá por encima de la línea principal y el resto de las opciones se mostrarán por debajo.



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



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

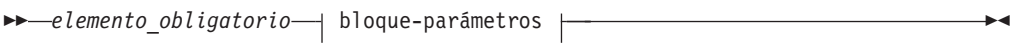


Una flecha de repetición por encima de una pila indica que puede elegir más de una opción de entre los elementos apilados o repetir una sola opción.

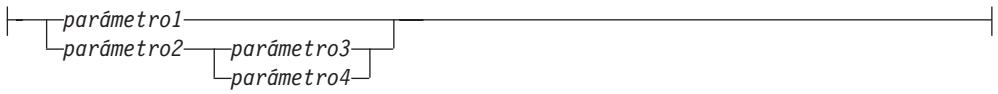
Las palabras clave aparecen en mayúsculas (por ejemplo, `FROM`). Deben escribirse exactamente igual a como aparecen en la sintaxis. Las variables aparecen en minúsculas (por ejemplo, `nombre-columna`). Representan nombres suministrados por el usuario o valores de la sintaxis.

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

A veces, una sola variable representa un fragmento mayor de la sintaxis. Por ejemplo, en el diagrama siguiente, la variable `bloque-parámetros` representa todo el fragmento de sintaxis denominado **bloque-parámetros**:



bloque-parámetros:



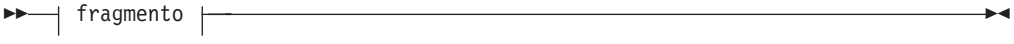
Los segmentos adyacentes que aparecen entre "puntos grandes" (●) se pueden especificar en cualquier secuencia.

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

```
elemento_obligatorio elemento1 elemento2 elemento3 elemento4
elemento_obligatorio elemento1 elemento3 elemento2 elemento4
```

Elementos de sintaxis comunes

Las secciones siguientes describen una serie de fragmentos de sintaxis que se utilizan en diagramas de sintaxis. Se hace referencia a los fragmentos de la forma siguiente:

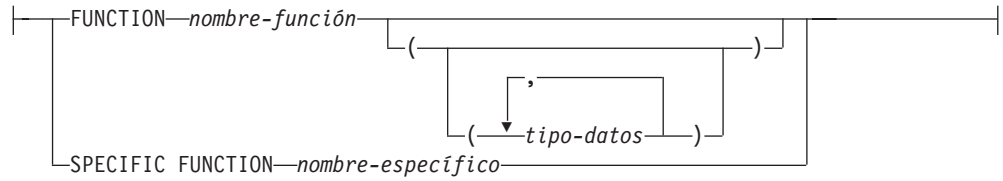


Designador de función

Un desginador de función identifica una sola función de forma exclusiva. Los designadores de función suelen aparecen en sentencias DDL para funciones (como, por ejemplo, DROP o ALTER).

Sintaxis:

designador-función:



Descripción:

FUNCTION *nombre-función*
Identifica una función específica y sólo es válido si hay exactamente una instancia de función con el nombre *nombre-función* en el esquema. La función identificada puede tener un número cualquiera de parámetros definidos para ella.En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/vinculación 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 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. El algoritmo de resolución de funciones no se utiliza.

nombre-función
Especifica el nombre de la función. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/vinculación 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. El número de tipos de datos y la concatenación lógica de los tipos de datos se utiliza para identificar la instancia de función específica.

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

Para los tipos de datos parametrizados no es necesario especificar la longitud, la precisión ni la escala. En su lugar, se puede codificar un conjunto vacío de paréntesis para indicar que estos atributos se han de pasar por alto al buscar coincidencias de un tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601), porque 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*, porque 0 < *n* < 25 significa REAL y 24< *n* < 54 significa DOUBLE. La coincidencia se produce en base de 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).

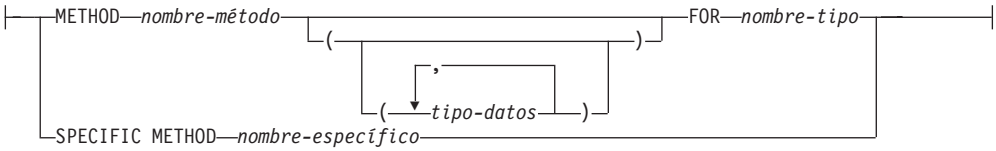
SPECIFIC FUNCTION *nombre-específico*
Identifica una función definida por el usuario concreta, mediante un nombre que se especifica o se toma por omisión al crear la función. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/vinculación 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 desginador de método identifica un solo método de forma exclusiva. Los designadores de método suelen aparecen en sentencias DDL para métodos (como, por ejemplo, DROP o ALTER).

Sintaxis:

designador-método:



Descripción:

METHOD *nombre-método*
Identifica un método determinado y sólo es válido si hay exactamente una instancia de método cuyo nombre sea *nombre-método* y su tipo sea *nombre-tipo*. El método identificado puede tener un número cualquiera de parámetros definidos para él. Si no existe ningún método con este nombre para el tipo, se produce un error (SQLSTATE 42704). Si existe más de una instancia específica del método para el tipo, se produce un error (SQLSTATE 42725).

METHOD *nombre-método (tipo-datos,...)*
Proporciona la signatura del método, que identifica de manera exclusiva el método. El algoritmo de resolución de métodos no se utiliza.

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. El número de tipos de datos y la concatenación lógica de los tipos de datos se utiliza para identificar la instancia de método específica.

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

Para los tipos de datos parametrizados no es necesario especificar la longitud, la precisión ni la escala. En su lugar, se puede codificar un conjunto vacío de paréntesis para indicar que estos atributos se han de pasar por alto al buscar coincidencias de un tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601), porque 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 TYPE.

No es necesario que un tipo de 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 en base de si el tipo es REAL o DOUBLE.

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

FOR *nombre-tipo*
Designa el tipo con el cual se debe asociar el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/vinculación QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados.

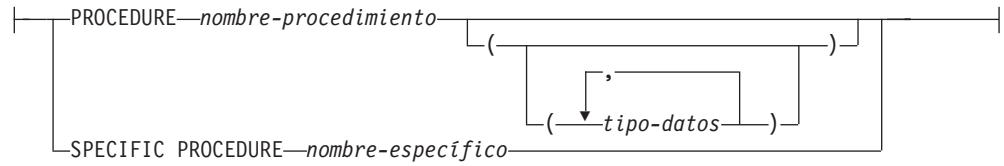
SPECIFIC METHOD *nombre-específico*
Identifica un método concreto, mediante el nombre que se especifica o se toma por omisión al crear el método. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/vinculación 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 nombrado o implícito; de lo contrario, se produce un error (SQLSTATE 42704).

Designador de procedimiento

Un desginador de procedimiento identifica un solo procedimiento almacenado de forma exclusiva. Los designadores de procedimiento suelen aparecen en sentencias DDL para procedimientos (como, por ejemplo, DROP o ALTER).

Sintaxis:

designador-procedimiento:



Descripción:

PROCEDURE *nombre-procedimiento*
Identifica un procedimiento concreto y sólo es válido si hay exactamente una instancia de procedimiento con el nombre *nombre-procedimiento* en el esquema. El procedimiento identificado puede tener un número cualquiera de parámetros definidos para él. En las sentencias de SQL dinámicas, el registro

especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/vinculación 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 mencionado o implicado, se producirá un error (SQLSTATE 42725).

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

nombre-procedimiento
Especifica el nombre del procedimiento. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/vinculación 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. El número de tipos de datos y la concatenación lógica de los tipos de datos se utiliza para identificar la instancia de procedimiento específica.

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

Para los tipos de datos parametrizados no es necesario especificar la longitud, la precisión ni la escala. En su lugar, se puede codificar un conjunto vacío de paréntesis para indicar que estos atributos se han de pasar por alto al buscar coincidencias de un tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601), porque 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 PROCEDURE.

No es necesario que un tipo de 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 en base de 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 concreto, mediante el nombre que se especifica o se toma por omisión al crear el procedimiento. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/vinculación QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia del procedimiento específico en el esquema nombrado o implícito; de lo contrario, se genera un error (SQLSTATE 42704).

Convenios utilizados en este manual

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

Condiciones de error

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

```
Si hay una
signatura duplicada se genera un error de SQL
(SQLSTATE 42723).
```

Convenios de resaltado

Se utilizan los siguientes convenios en este manual.

Negrita	Indica mandatos, palabras clave y otros elementos cuyos nombres están predefinidos por el sistema.
Cursiva	Indica uno de los casos 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 casos 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 concretos. • Ejemplos de texto similar al que puede mostrar el sistema. • Ejemplos de mensajes del sistema.

Documentación relacionada

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

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

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

Capítulo 1. Conceptos

En este capítulo se proporciona una vista de alto nivel de los conceptos que son importantes para comprender cuándo utilizar el Lenguaje de consulta estructurada (SQL). El material de consulta contenido en el resto de este manual proporciona una vista más detallada.

Bases de datos relacionales

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

Una base de datos relacional *particionada* es una base de datos relacional cuyos datos se gestionan repartidos en múltiples particiones (también denominadas nodos). Esta separación de los datos entre particiones es transparente para los usuarios de la mayoría de sentencias de SQL. Sin embargo, algunas sentencias DLL (lenguaje de definición de datos) tienen en cuenta la información de las particiones (por ejemplo, CREATE DATABASE PARTITION GROUP). (DLL, lenguaje de definición de datos, es el subconjunto de sentencias de SQL que se utilizan para describir las relaciones de los datos de una base de datos.)

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

Lenguaje de consulta estructurada (SQL)

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

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

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

Privilegios, niveles de autorización y autorizaciones sobre bases de datos

Los *privilegios* permiten a los usuarios crear recursos de la base de datos o acceder a los mismos. Los *niveles de autorización* proporcionan un método para agrupar los privilegios y las operaciones de mantenimiento y de programas de utilidad de nivel superior del gestor de bases de datos. Las *autorizaciones sobre bases de datos* permiten a los usuarios realizar actividades a nivel de la base de datos. Los privilegios, los niveles de autorización y las autorizaciones sobre bases de datos pueden utilizarse en conjunto para controlar el acceso al gestor de bases de datos y a los objetos del mismo. Los usuarios sólo puede acceder a los objetos sobre los que cuenten con el privilegio, el nivel de autorización o la autorización sobre bases de datos que se necesita y que DB2® Universal Database (DB2 UDB) determina cuando realiza una comprobación de autorización de un usuario autentificado.

El gestor de bases de datos necesita que todos los usuarios estén específicamente autorizados, ya sea implícita o explícitamente, para utilizar cada una de las funciones de la base de datos necesarias para realizar una tarea específica. Las autorizaciones o los privilegios *explícitos* se otorgan al usuario (GRANTEETYPE de U en los catálogos de bases de datos). Las autorizaciones o los privilegios *implícitos* se otorgan a un grupo al que pertenece el usuario (GRANTEETYPE de G en los catálogos de bases de datos). De este modo, para crear una tabla, un usuario debe tener autorización para crear tablas; para modificar una tabla, un usuario debe tener autorización para modificar tablas y así sucesivamente.

La Figura 1 muestra la relación entre las autorizaciones y su ámbito de control (para bases de datos, gestores de bases de datos).

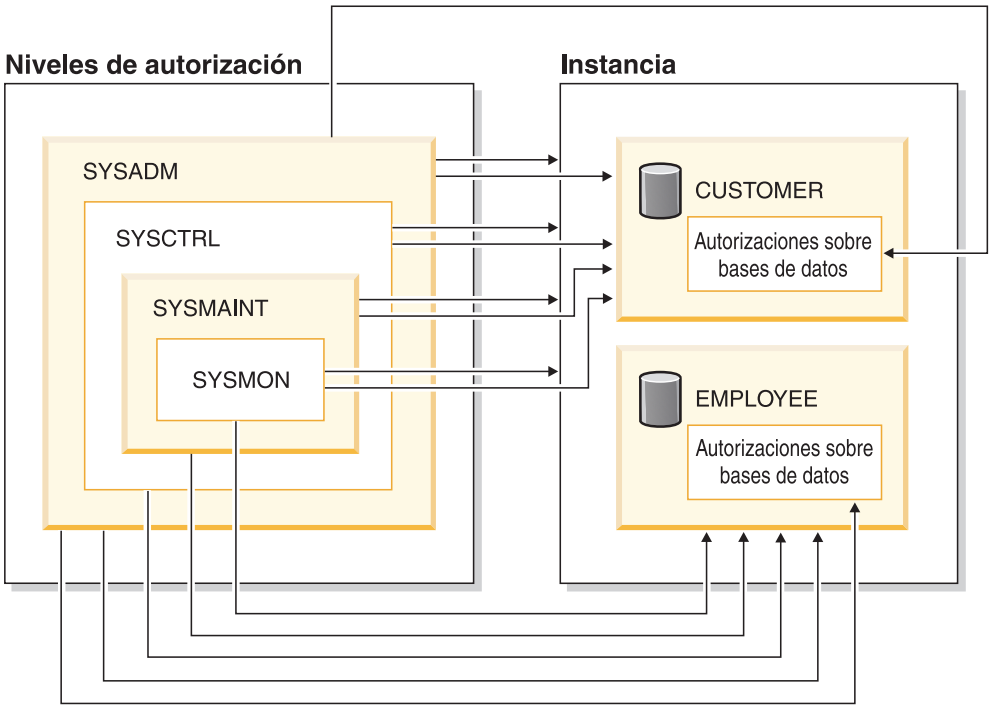


Figura 1. Jerarquía de las autorizaciones

Un usuario o grupo puede tener una o varias de las autorizaciones o los privilegios siguientes:

- Autorización de administrador:
  - SYSADM (administrador del sistema)

    El nivel de autorización SYSADM proporciona control sobre todos los recursos que el gestor de bases de datos crea y mantiene. El administrador del sistema posee todas las autorizaciones DBADM, SYSCTRL, SYSMAINT y SYSMON, además de la autorización de otorgar y revocar la autorización DBADM.

    El usuario que posee la autorización SYSADM es responsable de controlar el gestor de bases de datos y de garantizar la seguridad y la integridad de los datos. La autorización SYSADM proporciona privilegios implícitos sobre todos los objetos de la base de datos, control sobre qué usuarios pueden acceder al gestor de bases de datos y el ámbito de este acceso. Para obtener información adicional sobre la autorización SYSADM, consulte ″Autorización de administrador del sistema (SYSADM)″.
  - DBADM (administrador de la base de datos)

    La autorización sobre bases de datos DBADM proporciona autorización para administrar una sola base de datos. Este administrador de la base de datos posee los privilegios necesarios para crear objetos, emitir mandatos de la base de datos y acceder a datos de las tablas. El administrador de la base de datos también puede otorgar y revocar el privilegio CONTROL y otros privilegios individuales. Para obtener información adicional sobre la autorización DBADM, consulte ″Autorización de administrador de la base de datos(DBADM)″.
- Autorización de control del sistema:
  - SYSCTRL (control del sistema)

    El nivel de autorización SYSCTRL proporciona control sobre las operaciones que afectan a los recursos del sistema. Por ejemplo, un usuario con autorización SYSCTRL puede crear, actualizar, detener o descartar una base de datos. Este usuario también puede detener una instancia pero no acceder a los datos de las tablas. Los usuarios con autorización SYSCTRL también poseen autorización SYSMON. Para obtener información adicional acerca de la autorización SYSCTRL, consulte ″Autorización de control del sistema(SYSCTRL)″.
  - SYSMAINT (mantenimiento del sistema)

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

  El nivel de autorización SYSMON proporciona la autorización necesaria para utilizar el supervisor del sistema de bases de datos. Para obtener información adicional acerca de la autorización SYSMON, consulte ″Autorización de supervisor del sistema(SYSMON)″
- Autorizaciones sobre bases de datos

Para realizar actividades como, por ejemplo, crear una tabla o una rutina, o para cargar datos en una tabla, se necesitan autorizaciones específicas sobre las bases de datos. Para obtener información adicional, consulte ″Autorizaciones sobre bases de datos″.

- Privilegios:

  Los privilegios se necesitan para realizar actividades sobre objetos de la base de datos (por ejemplo, para crear y descartar un índice). Los privilegios definen de forma precisa las tareas que un usuario puede realizar. Por ejemplo, un usuario puede tener el privilegio para crear un índice en una tabla pero no un activador en la misma tabla.

  – Privilegio CONTROL

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

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

    Si un usuario distinto necesita el privilegio CONTROL sobre este objeto, un usuario con autorización SYSADM o DBADM debe otorgarle el privilegio CONTROL sobre dicho objeto. El privilegio CONTROL no puede revocarse del propietario del objeto.

    En algunas situaciones, el creador de un objeto obtiene el privilegio CONTROL sobre el objeto de forma automática. Para obtener información adicional, consulte ″Creación, posesión y privilegios de un objetos″.

  – Es posible otorgar privilegios individuales que permiten a un usuario llevar a cabo determinadas tareas sobre objetos concretos.

    Los usuarios con autorización de administrador (SYSADM o DBADM) o el privilegio CONTROL pueden otorgar privilegios a los usuarios y revocarlos.

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

    Los privilegios también pueden otorgarse a PUBLIC. Los privilegios PUBLIC son aplicables a todos los usuarios(nombres de autorización), incluidos los usuarios futuros, con independencia de si a algún usuario individual se le ha otorgado el privilegio anteriormente.

  – Los privilegios implícitos pueden otorgarse a un usuario que tenga el privilegio a ejecutar un paquete. Los usuarios pueden ejecutar la aplicación pero no necesitan obligatoriamente privilegios explícitos sobre los objetos de los datos utilizados en el paquete.

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

Nota: Debe tenerse cuidado cuando se otorgan autorizaciones y privilegios a un nombre de autorización y no se ha creado ningún usuario con este nombre de autorización. Más tarde, puede crearse un usuario con este nombre de autorización y recibiría todas las autorizaciones y privilegios asociados con este nombre de autorización.

La sentencia REVOKE se utiliza para revocar los privilegios otorgados con anterioridad. En DB2 UDB, al revocar un privilegio de un nombre de autorización se revoca el privilegio otorgado por todos los nombres de autorización.

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

Conceptos relacionados:

- "System administration authority (SYSADM)" en la publicación *Administration Guide: Implementation*
- "System control authority (SYSCTRL)" en la publicación *Administration Guide: Implementation*
- "System maintenance authority (SYSMAINT)" en la publicación *Administration Guide: Implementation*
- "Database administration authority (DBADM)" en la publicación *Administration Guide: Implementation*
- "LOAD authority" en la publicación *Administration Guide: Implementation*
- "Database authorities" en la publicación *Administration Guide: Implementation*
- "Schema privileges" en la publicación *Administration Guide: Implementation*
- "Table space privileges" en la publicación *Administration Guide: Implementation*
- "Table and view privileges" en la publicación *Administration Guide: Implementation*
- "Package privileges" en la publicación *Administration Guide: Implementation*
- "Index privileges" en la publicación *Administration Guide: Implementation*
- "Sequence privileges" en la publicación *Administration Guide: Implementation*
- "Controlling access to database objects" en la publicación *Administration Guide: Implementation*
- "Indirect privileges through a package" en la publicación *Administration Guide: Implementation*
- "Routine privileges" en la publicación *Administration Guide: Implementation*
- "Object creation, ownership, and privileges" en la publicación *Administration Guide: Implementation*
- "System monitor authority (SYSMON)" en la publicación *Administration Guide: Implementation*

Esquemas

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

Esquemas

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

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

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

```
CREATE SCHEMA C AUTHORIZATION A
```

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

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

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

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

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

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

Conceptos relacionados:

- "Schema privileges" en la publicación *Administration Guide: Implementation*

Tablas

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

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

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

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

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

Vistas

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

Es posible utilizar las vistas para controlar el acceso a datos sensibles, porque las vistas permiten que muchos usuarios vean presentaciones distintas de los mismos datos. Por ejemplo, es posible que varios usuarios accedan a un tabla de datos sobre los empleados. Un director ve los datos sobre sus empleados pero no de los empleados de otro departamento. Un oficial de reclutamiento ve las fechas de contratación de todos los empleados, pero no sus salarios; un oficial de finanzas ve los salarios, pero no las fechas de contratación. Cada uno de estos usuarios trabaja con una vista derivada de la tabla base. Cada vista se parece a una tabla y tiene nombre propio.

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

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

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

Seudónimos

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

Al igual que las tablas o las vistas, un seudónimo puede crearse, eliminarse y tener comentarios asociados. Sin embargo, a diferencia de las tablas, los seudónimos pueden hacerse referencia entre sí en un proceso llamado *encadenamiento*. Los seudónimos son nombres de referencia pública, por lo que no es necesaria ninguna autorización ni privilegio especial para utilizarlos. Sin embargo, el acceso a la tabla o a la vista a la que un seudónimo hace referencia sí que requiere la autorización asociada con estos objetos.

Hay otros tipos de seudónimos como, por ejemplo, los seudónimos de base de datos y de red. También se pueden crear seudónimos para *apodos* que hagan referencia a vistas o tablas de datos ubicadas en sistemas federados.

Índices

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

El gestor de bases de datos utiliza los índices para:

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

Claves

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

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

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

Una *clave primaria* es un caso especial de clave de unicidad. Una tabla no puede tener más de una clave primaria.

Una *clave foránea* es una clave que se especifica en la definición de una restricción de referencia.

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

Restricciones

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

Existen cuatro tipos de restricciones:

- Una *restricción de unicidad* es una regla que prohíbe los valores duplicados en una o varias columnas de una tabla. Las restricciones de unicidad a las que se da soporte son la clave de unicidad y la clave primaria. Por ejemplo, se puede definir una restricción de unicidad en el identificador de proveedor de la tabla de proveedores para asegurarse de que no se da el mismo identificador de proveedor a dos proveedores.
- Una *restricción de referencia* es una regla lógica acerca de los valores de una o varias columnas de una o varias tablas. Por ejemplo, un conjunto de tablas que comparte información sobre los proveedores de una empresa. Ocasionalmente, el nombre de un proveedor cambia. Puede definir una restricción de referencia que indique que el ID del proveedor de una tabla debe coincidir con un ID de proveedor de la información de proveedor. Esta restricción impide que se realicen operaciones de inserción, actualización o supresión que, de lo contrario, harían que faltara información del proveedor.
- Una *restricción de comprobación de tabla* establece restricciones en los datos que se añaden a una tabla específica. Por ejemplo, una restricción de comprobación de tabla puede garantizar que el nivel salarial de un empleado no sea inferior a 20.000 euros siempre que se añadan o se actualicen datos salariales en una tabla que contiene información de personal.
- Una *restricción informativa* es una regla que el compilador de SQL puede utilizar pero que no viene impuesta por el gestor de bases de datos.

Las restricciones de referencia y de comprobación de tabla pueden activarse y desactivarse. Generalmente es una buena idea, por ejemplo, desactivar la imposición de una restricción cuando se cargan grandes cantidades de datos en una base de datos.

Restricciones de unicidad

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

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

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

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

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

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

Restricciones de referencia

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

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

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

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

Las restricciones de referencia con una regla de supresión o una regla de actualización de RESTRICT se imponen antes que el resto de restricciones de referencia. Las restricciones de referencia con una regla de supresión o una regla de actualización de NO ACTION se comportan igual que RESTRICT en la mayoría de casos.

Tenga en cuenta que es posible combinar las restricciones de referencia, las restricciones de comprobación y los activadores.

Las reglas de integridad de referencia implican los conceptos y terminología siguientes:

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

Fila padre
Fila que tiene, como mínimo, una fila dependiente.

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

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

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

Fila dependiente
Fila que tiene, como mínimo, una fila padre.

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

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

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

Fila de autorreferencia
Fila que es padre de sí misma.

Regla de inserción

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

Regla de actualización

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

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

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

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

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

Regla de supresión

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

La regla de supresión de una restricción de referencia se aplica al suprimir una fila de la tabla padre. Para ser más exactos, esta regla se aplica cuando una fila de la tabla padre es el objeto de una operación de supresión o de supresión propagada (definida a continuación) y dicha fila tiene dependientes en la tabla dependiente de la restricción de referencia. Examinemos un ejemplo donde P es la tabla padre, D es la tabla dependiente y p es una fila padre que es el objeto de una operación de supresión o de supresión propagada. La regla de supresión funciona del modo siguiente:

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

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

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

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

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

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

Las restricciones siguientes son aplicables a las relaciones conectadas por supresión:

- Cuando una tabla está conectada por supresión con sigo mismo en un ciclo referencial de más de una tabla, el ciclo no debe contener una regla de supresión RESTRICT o SET NULL.

- Una tabla no debe ser una tabla dependiente en una relación de tipo CASCADE (con referencia a sí misma o a otra tabla) y tener una relación de referencia a sí misma con una regla de supresión RESTRICT o SET NULL.
- Cuando una tabla está conectada por supresión con otra tabla a través de varias relaciones y éstas poseen claves foráneas de solapamiento, dichas relaciones deben tener la misma regla de supresión y ninguna de éstas puede ser SET NULL.
- Cuando una tabla está conectada por supresión con otra tabla a través de varias relaciones y una de las relaciones se especifica con la regla de supresión SET NULL, la definición de clave foránea de dicha relación no debe contener ninguna clave de particionamiento ni ninguna columna de clave MDC.
- 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 en caso de que las vías de acceso conectadas por supresión terminen con la regla de supresión RESTRICT o SET NULL.

Restricciones de comprobación de tabla

Una *restricción de comprobación de tabla* es una regla que especifica los valores permitidos en una o varias columnas de cada fila de una tabla. Una restricción es opcional y puede definirse utilizando la sentencia CREATE TABLE o ALTER TABLE. La especificación de restricciones de comprobación de tabla se realiza mediante una forma restringida de condición de búsqueda. Una de las restricciones consiste en que un nombre de columna de una restricción de comprobación de tabla de la tabla T debe identificar una columna de la tabla T.

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

Cuando hay una o varias restricciones de comprobación de tabla definidas en la sentencia ALTER TABLE para una tabla en la que existen datos, éstos se comprueban con la nueva condición antes de que finalice la sentencia ALTER TABLE. La sentencia SET INTEGRITY puede utilizarse para colocar la tabla en estado *pendiente de comprobación*, lo que permite que la sentencia ALTER TABLE prosiga sin comprobar los datos.

Restricciones informativas

Una *restricción informativa* es una regla que el compilador de SQL puede utilizar para mejorar la vía de acceso a los datos. Las restricciones informativas no vienen impuestas por el gestor de bases de datos y no se utilizan para verificar los datos de forma adicional. Simplemente se utilizan para mejorar el rendimiento de la consulta.

Para definir una restricción referencial o de comprobación de tabla, debe utilizarse la sentencia CREATE TABLE o ALTER TABLE, especificando los atributos de restricción que determinen si el gestor de bases de datos debe o no debe imponer la restricción y si la restricción debe o no debe utilizarse para la optimización de las consultas.

Información relacionada:

- "Sentencia SET INTEGRITY" en la publicación *Consulta de SQL, Volumen 2*
- Apéndice H, "Interacción de los activadores con las restricciones", en la página 745

Niveles de aislamiento

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

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

El nivel de aislamiento correspondiente a sentencias de SQL estático se especifica como un atributo de un paquete y se aplica a los procesos de aplicación que utilizan el paquete. El nivel de aislamiento se especifica en el proceso de preparación del proceso. Para sentencias de SQL dinámico, el nivel de aislamiento por omisión es el nivel de aislamiento especificado para el paquete que prepara la sentencia. La sentencia SET CURRENT ISOLATION permite especificar niveles de aislamiento alternativos para SQL emitido dentro de una sesión. En función del tipo de bloqueo, limita o impide el acceso a los datos por parte de procesos de aplicación simultáneos. (Las tablas temporales declaradas y las filas de las mismas no pueden bloquearse, pues sólo la aplicación que las declaró puede acceder a ellas).

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

Compartimiento
Limita los procesos de aplicación simultáneos a operaciones de sólo lectura de los datos.

Actualización
Limita los procesos de aplicación simultáneos a operaciones de sólo lectura de los datos, si dichos procesos no han declarado que podrían actualizar la fila. El gestor de bases de datos asume que el proceso que actualmente mira a una fila es posible que la actualice.

Exclusivo
Evita que los procesos de aplicación simultáneos accedan a los datos de todas formas. No se aplica a los procesos de aplicación con un nivel de aislamiento de *lectura no confirmada*, que pueden leer los datos pero no modificarlos.

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

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

- Lectura repetible (RR)

  Este nivel garantiza que:

- Cualquier fila leída durante una unidad de trabajo no puede modificarla ningún otro proceso de aplicación hasta que la unidad de trabajo haya finalizado. Las filas se leen en la misma unidad de trabajo que la sentencia OPEN correspondiente. El uso de la cláusula WITH RELEASE opcional en la sentencia CLOSE significa que, si se vuelve a abrir el cursor, ya no se aplicará ninguna garantía respecto a las lecturas no repetibles y no se aplicarán ya lecturas fantasma a ninguna fila a la que se haya accedido anteriormente.
- Las filas modificadas por otro proceso de aplicación no se pueden leer hasta que dicho proceso de aplicación las confirme.

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

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

- Estabilidad de lectura (RS)

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

  - Cualquier fila leída durante una unidad de trabajo no puede modificarla ningún otro proceso de aplicación hasta que la unidad de trabajo haya finalizado. Las filas se leen en la misma unidad de trabajo que la sentencia OPEN correspondiente. El uso de la cláusula WITH RELEASE opcional en la sentencia CLOSE significa que, si se vuelve a abrir el cursor, ya no se aplicará ninguna garantía respecto a las lecturas no repetibles a ninguna fila a la que se haya accedido anteriormente.
  - Las filas modificadas por otro proceso de aplicación no se pueden leer hasta que dicho proceso de aplicación las confirme.

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

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

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

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

- Estabilidad del cursor (CS)

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

  A diferencia de la Lectura repetible, la Estabilidad del cursor sólo asegura que otros procesos de aplicación no modifiquen la fila actual de cada cursor

actualizable. De este modo, las filas leídas durante una unidad de trabajo pueden ser modificadas por otros procesos de aplicación.

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

- Lectura no confirmada (UR)

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

  – Cualquier fila leída durante una unidad de trabajo sea modificada por otros procesos de aplicación.
  – Cualquier fila cambiada por otro proceso de aplicación pueda leerse aunque dicho proceso de aplicación no ha confirmado el cambio.

  Para otras operaciones, se aplican las reglas asociadas con el nivel CS.

Comparación de niveles de aislamiento

La tabla siguiente resume la información acerca de los niveles de aislamiento.

	UR	**CS**	**RS**	**RR**
¿La aplicación puede ver los cambios no confirmados realizados por otros procesos de aplicación?	Sí	No	No	No
¿La aplicación puede actualizar los cambios no confirmados realizados por otros procesos de aplicación?	No	No	No	No
¿Pueden otros procesos de aplicación afectar a la operación de volver a ejecutar una sentencia? *Consulte el fenómeno P3 (fantasma) al final de la tabla.*	Sí	Sí	Sí	No
¿Pueden otros procesos de aplicación actualizar filas “actualizadas”? Consulte la Nota 1 al final de la tabla.	No	No	No	No
¿Pueden otros procesos de aplicación que se ejecutan a un nivel de aislamiento que no sea UR leer las filas “actualizadas”?	No	No	No	No
¿Pueden otros procesos de aplicación que se ejecutan al nivel de aislamiento UR leer las filas “actualizadas”?	Sí	Sí	Sí	Sí
¿Pueden otros procesos de aplicación actualizar las filas “a las que se ha accedido”? *Consulte el fenómeno P2 (lectura no repetible) al final de la tabla.*	Sí	Sí	No	No
¿Pueden otros procesos de aplicación leer las filas “a las que se ha accedido”?	Sí	Sí	Sí	Sí
¿Pueden otros procesos de aplicación actualizar o suprimir la fila “actual”? *Consulte el fenómeno P1 (lectura sucia) al final de la tabla.*	Consulte la nota 2 al final de la tabla.	Consulte la nota 2 al final de la tabla.	No	No

	UR	CS	RS	RR

Notas:

1. El nivel de aislamiento no ofrece protección a la aplicación si ésta lee y escribe una tabla. Por ejemplo, una aplicación abre un cursor en una tabla y entonces realiza una operación de inserción, actualización o supresión en la misma tabla. Es posible que la aplicación vea datos incoherentes al buscar más filas desde el cursor abierto.
2. Si el cursor no es actualizable, con CS la fila actual puede actualizarse o suprimirse por otros procesos de aplicación en algunos casos. Por ejemplo, el almacenamiento intermedio puede hacer que la fila actual del cliente sea distinta de la fila actual que realmente aparece en el servidor.

Ejemplos de fenómenos:

P1 *Lectura sucia.* La unidad de trabajo UW1 modifica una fila. La unidad de trabajo UW2 lee dicha fila antes que UW1 ejecute COMMIT. Si UW1 realiza entonces una operación ROLLBACK, UW2 ha leído una fila no existente.

P2 *Lectura no repetible.* La unidad de trabajo UW1 lee una fila. La unidad de trabajo UW2 modifica dicha fila y realiza una operación COMMIT. Si UW1 vuelve a leer la fila, puede recibir un valor modificado.

P3 *Fantasma.* La unidad de trabajo UW1 lee el conjunto de *n* filas que satisface alguna condición de búsqueda.La unidad de trabajo UW2 inserta (INSERT) una o varias filas que satisfacen la condición de búsqueda y la confirma (COMMIT). Si UW1 repite la lectura inicial con la misma condición de búsqueda, obtiene las filas originales más las filas insertadas.

Información relacionada:

- "Sentencia DECLARE CURSOR" en la publicación *Consulta de SQL, Volumen 2*

Consultas y expresiones de tabla

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

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

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

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

Información relacionada:

- "Sentencia select" en la página 513
- "Consultas de SQL" en la página 469

Procesos, simultaneidad y recuperación de aplicaciones

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

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

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

El gestor de bases de datos proporciona una forma de restituir los cambios no confirmados realizados por un proceso de aplicación. Esto podría ser necesario en caso de error en un proceso de aplicación o si se produce un punto muerto o un tiempo excedido por bloqueo. Un proceso de aplicación puede solicitar de modo explícito que se restituyan los cambios en la base de datos. Esto se hace utilizando una operación de *retrotracción*.

Una *unidad de trabajo* es una secuencia recuperable de operaciones dentro de un proceso de aplicación. Una unidad de trabajo se inicia cuando se inicia un proceso de aplicación o cuando termina la unidad de trabajo anterior y no sea debido a la finalización del proceso de aplicación. Una unidad de trabajo finaliza mediante una operación de confirmación, de retrotracción o por el final de un proceso de aplicación. Una operación de retrotracción o confirmación sólo afecta a los cambios realizados en la base de datos durante la unidad de trabajo que está terminando.

Mientras estos cambios permanecen sin confirmar, otros procesos de aplicaciones no pueden percibirlos y dichos cambios pueden restituirse. Sin embargo, esto no es así cuando el nivel de aislamiento es de lectura no confirmada (UR). Una vez confirmados, los demás procesos de aplicación pueden acceder a estos cambios de la base de datos y éstos ya no se pueden restituir mediante una retrotracción.

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

Los bloqueos adquiridos por el gestor de bases de datos para un proceso de aplicación se conservan hasta el final de una unidad de trabajo. Sin embargo, esto no es así cuando el nivel de aislamiento es de estabilidad del cursor (CS, en el que el bloqueo se libera cuando el cursor se mueve de una fila a otra) o de lectura no confirmada(UR, en el que no se obtienen bloqueos).

No se puede impedir nunca que un proceso de aplicación realice operaciones debido a sus propios bloqueos. Sin embargo, si una aplicación utiliza transacciones simultáneas, los bloqueos de una transacción pueden afectar la operación de una transacción simultánea.

El inicio y la finalización de una unidad de trabajo definen los puntos de coherencia en un proceso de aplicación. Por ejemplo, una transacción bancaria puede incluir la transferencia de fondos de una cuenta a otra. Una transacción de este tipo necesitaría que dichos fondos se restaran de la primera cuenta y se sumaran más tarde a la segunda cuenta. Después de esta sustracción, los datos son incoherentes. La coherencia sólo queda restablecida cuando los fondos se han sumado a la segunda cuenta. Cuando se hayan completado los dos pasos, podrá utilizarse la operación de confirmación para finalizar la unidad de trabajo, con lo que los cambios estarán disponibles para otros procesos de aplicación. Si se produce una anomalía antes de que finalice la unidad de trabajo, el gestor de bases de datos retrotraerá los cambios no confirmados para restablecer la coherencia de los datos que se presupone que existía al iniciar la unidad de trabajo.

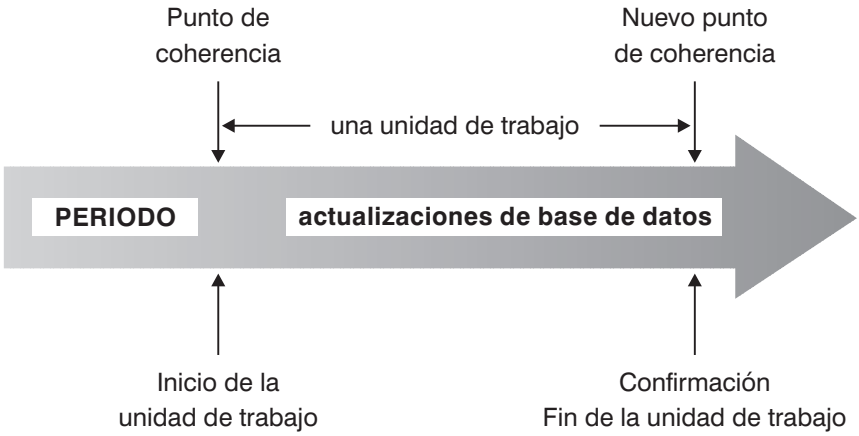


Figura 2. Unidad de trabajo con una sentencia COMMIT

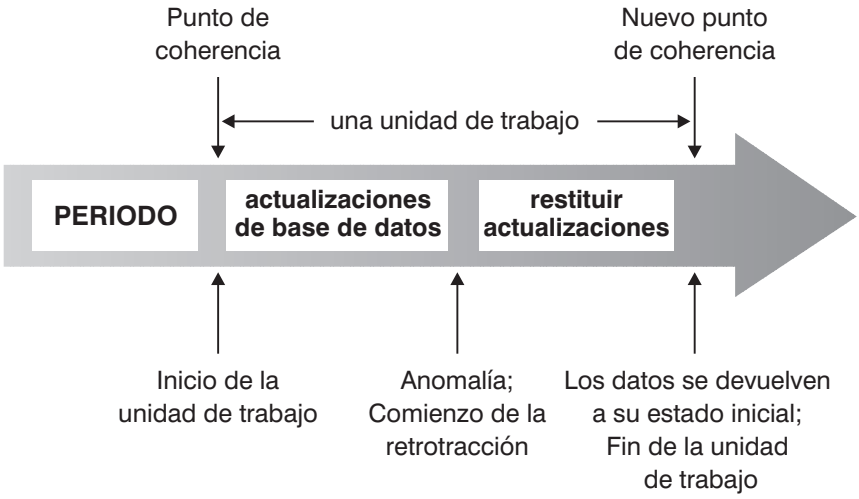


Figura 3. Unidad de trabajo con una sentencia ROLLBACK

Conceptos relacionados:

• "Niveles de aislamiento" en la página 15

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

La interfaz de nivel de llamada de DB2 es una interfaz de programación de aplicaciones que proporciona funciones para que los programas de aplicación procesen las sentencias de SQL dinámico. Los programas CLI también se pueden compilar utilizando el Software Developer's Kit de Open Database Connectivity (suministrado por Microsoft u otro proveedor), que permite acceder a fuentes de datos ODBC. A diferencia del SQL interno, esta interfaz no requiere precompilación. Las aplicaciones pueden ejecutarse en distintas bases de datos sin necesidad de volver a compilarlas en cada una de estas bases de datos. Las aplicaciones utilizan llamadas a procedimientos en tiempo de ejecución para conectarse a bases de datos, emitir sentencias de SQL y recuperar datos e información de estado.

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

- La CLI proporciona llamadas de función que soportan una forma de consultar catálogos de bases de datos que es coherente en toda la familia DB2. Esto reduce la necesidad de escribir consultas de catálogo que deban adaptarse a servidores de bases de datos concretos.
- La CLI proporciona la capacidad de desplazarse con un cursor de estas maneras:
  - Hacia adelante, una o más filas
  - Hacia atrás, una o más filas
  - Hacia adelante desde la primera fila, una o más filas
  - Hacia atrás desde la última fila, una o más filas
  - Desde una posición en el cursor almacenada previamente.
- Los procedimientos almacenados llamados desde programas de aplicación que se hayan escrito con la CLI pueden devolver conjuntos resultantes a esos programas.

Programas Conectividad de bases de datos Java (JDBC) y SQL incorporado para Java (SQLJ)

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

- Las llamadas JDBC se convierten en llamadas CLI de DB2 a través de métodos nativos Java. Las peticiones JDBC fluyen del DB2 cliente a través de la CLI de DB2 hasta el servidor de DB2. JDBC no puede utilizar SQL estático.
- Las aplicaciones SQLJ utilizan JDBC como base para tareas como conectarse a bases de datos y manejar errores de SQL, pero también pueden contener sentencias de SQL estático intercaladas en los archivos fuente SQLJ. Un archivo fuente SQLJ debe convertirse con el conversor SQLJ para que se pueda compilar el código Java resultante.

Paquetes

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

Vistas de catálogo

El gestor de bases de datos mantiene un conjunto de vistas tablas base que contienen información sobre los datos que se encuentran bajo su control. Estas vistas y tablas base se conocen en su conjunto como el *catálogo*. El catálogo contiene información acerca de la estructura lógica y física de los objetos de la base de datos como, por ejemplo, tablas, vistas, índices, paquetes y funciones. También contiene información estadística. El gestor de bases de datos garantiza que las descripciones del catálogo siempre sean precisas.

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

Información relacionada:

• "Vistas de catálogo del sistema" en la página 549

Conversión de caracteres

Una *serie* es una secuencia de bytes que puede representar caracteres. Todos los caracteres de una serie tienen una representación de codificación común. En algunos casos, puede ser necesario convertir estos caracteres a una representación de códigos diferente, conocida como *conversión de caracteres*. La conversión de caracteres, cuando es necesaria, es automática y, cuando es satisfactoria, es transparente para la aplicación.

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

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

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

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

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

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

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

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

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

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

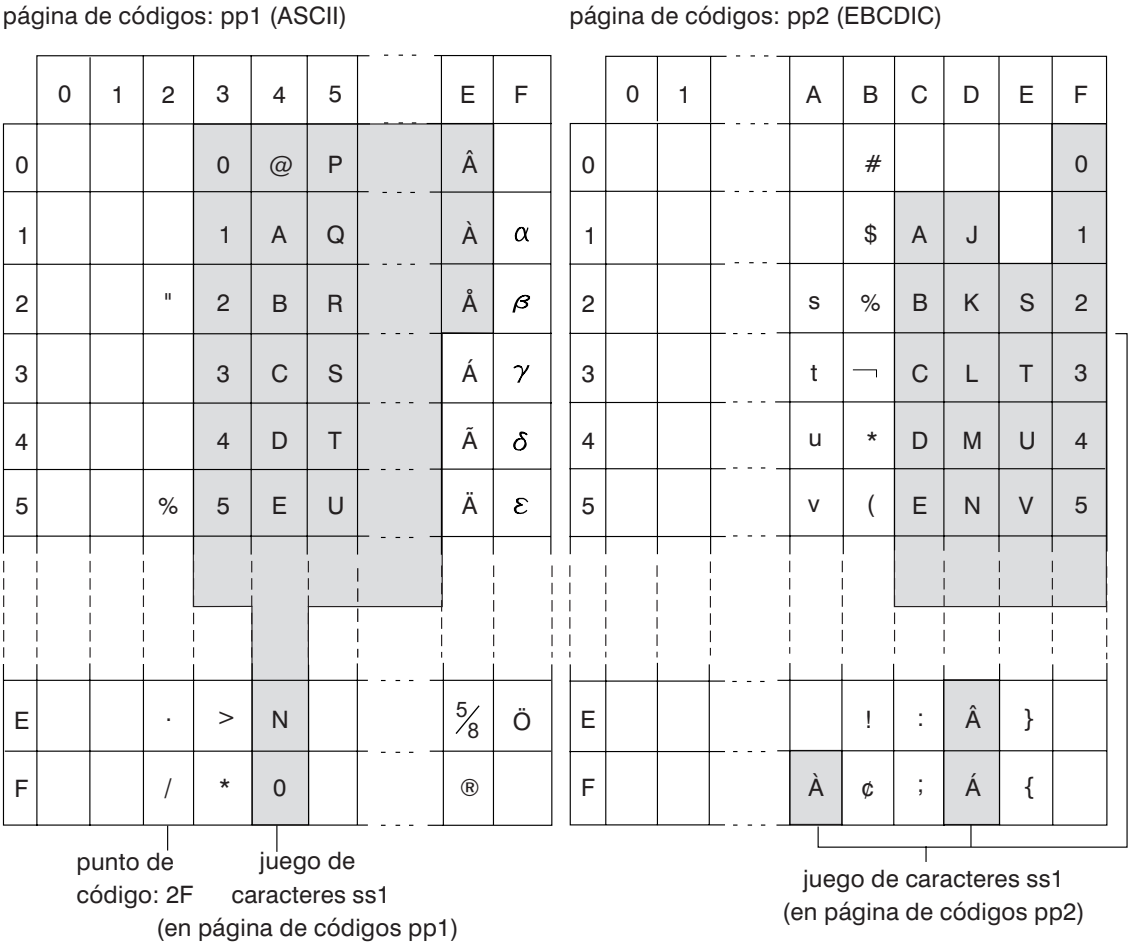


Figura 4. Correlación de un juego de caracteres en diferentes páginas de códigos

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

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

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

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

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

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

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

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

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

Información relacionada:

- "Asignaciones y comparaciones" en la página 110
- "Reglas para la conversión de series" en la página 129

Supervisores de sucesos

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

Nota: Por omisión, todas las bases de datos tienen definido un supervisor de sucesos denominado DB2DETAILDEADLOCK que realiza el seguimiento de DEADLOCKS WITH DETAILS. El supervisor de sucesos DB2DETAILDEADLOCK se inicia automáticamente al iniciar la base de datos.

Mientras el supervisor de instantáneas suele utilizarse para el mantenimiento preventivo y el análisis de problemas, los supervisores de sucesos se utilizan para alertar a los administradores sobre problemas inmediatos o para rastrear problemas inminentes.

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

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

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

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

La salida de los sucesos de eventos puede dirigirse a las tablas de SQL, a un archivo o a una área de interconexión con nombre.

Conceptos relacionados:

- "Database system monitor" en la publicación *System Monitor Guide and Reference*

Tareas relacionadas:

- "Collecting information about database system events" en la publicación *System Monitor Guide and Reference*
- "Creating an event monitor" en la publicación *System Monitor Guide and Reference*

Información relacionada:

- "Event monitor sample output" en la publicación *System Monitor Guide and Reference*
- "Event types" en la publicación *System Monitor Guide and Reference*

Activadores

Un *activador* define un conjunto de acciones que se ejecutan en respuesta a una operación de inserción, actualización o supresión en una tabla determinada. Cuando se ejecuta una de estas operaciones de SQL, se dice que el activador se ha *activado*.

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

Los activadores pueden utilizarse, junto con las restricciones de referencia y las restricciones de comprobación, para imponer las reglas de integridad de los datos. Los activadores también pueden utilizarse para provocar actualizaciones en otras

tablas, para transformar o generar valores automáticamente en las filas insertadas o actualizadas, o para invocar funciones que realicen tareas como la de emitir alertas.

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

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

Los elementos siguientes se especifican al crear un activador:

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

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

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

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

Otra forma de hacer referencia a los valores del conjunto de filas afectadas consisten en utilizar *tablas de transición*. Las tablas de transición también utilizan los nombres de las columnas de la tabla sujeto pero especifican un nombre para permitir que el conjunto completo de filas afectadas se trate como una tabla. Las tablas de transición sólo se pueden utilizar en activadores posteriores y es posible definir tablas de transición independientes para los valores anteriores y los nuevos.

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

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

Información relacionada:

- Apéndice H, “Interacción de los activadores con las restricciones”, en la página 745

Espacios de tablas y otras estructuras de almacenamiento

Las estructuras de almacenamiento contienen objetos de base de datos. La estructura básica de almacenamiento es el *espacio de tablas*; contiene tablas, índices, objetos grandes y datos definidos con un tipo de datos LONG. Existen dos tipos de espacios de tablas:

Espacio gestionado por base de datos (DMS)
Espacio de tablas que está gestionado por el gestor de bases de datos.

Espacio gestionado por el sistema (SMS)
Espacio de tablas que está gestionado por el sistema operativo.

Todos los espacios de tablas constan de contenedores. Un *contenedor* describe dónde se almacenan los objetos. Un subdirectorio de un sistema de archivos es un ejemplo de contenedor.

Cuando los datos se leen de los contenedores de espacios de tablas, se colocan en un área de la memoria denominada agrupación de almacenamientos intermedios. Una *agrupación de almacenamientos intermedios* está asociada a un espacio de tablas concreto y, por lo tanto, puede controlar qué datos compartirán las mismas áreas de memoria para el almacenamiento intermedio de datos.

En una base de datos particionada, los datos se reparten entre distintas particiones de base de datos. El grupo de particiones de base de datos asignado al espacio de tablas determina qué particiones en concreto se incluyen. Un *grupo de particiones de base de datos* es un grupo de una o varias particiones definidas como parte de la base de datos. Un espacio de tablas incluye uno o varios contenedores para cada partición del grupo de particiones de base de datos. El gestor de bases de datos utiliza un *mapa de particionamiento*, asociado con cada grupo de particiones de base de datos, para determinar en qué partición se almacenará un fila de datos en concreto. El mapa de particionamiento es una matriz de 4096 números de partición. El índice del mapa de particionamiento generado por la función de particionamiento para cada fila de una tabla se utiliza para determinar la partición en la que se almacenará una fila. A modo de ejemplo, la figura siguiente muestra cómo se correlaciona una fila con el valor de clave de particionamiento (c1, c2, c3) con un índice de mapa de particionamiento 2 que, a su vez, hace referencia a la partición p5.

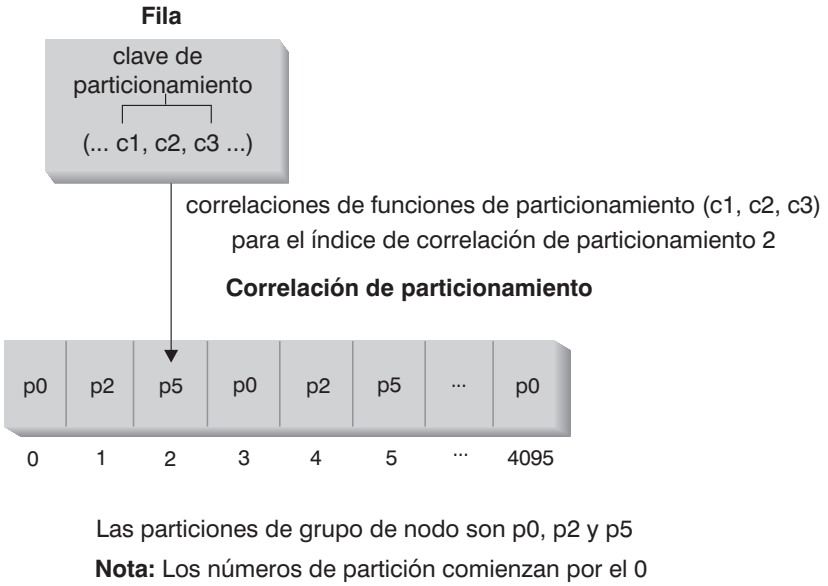


Figura 5. Distribución de datos

El mapa de particionamiento se puede cambiar, permitiendo cambiar la distribución de los datos sin modificar la clave de particionamiento ni los datos reales. El nuevo mapa de particionamiento se especifica como parte del mandato REDISTRIBUTE DATABASE PARTITION GROUP o de la interfaz de programación de aplicaciones (API) sqludrdt, que lo utilizan para redistribuir las tablas en el grupo de particiones de base de datos.

El producto DB2 Data Links Manager proporciona funciones que permiten posibilidades de almacenamiento adicionales. Una tabla de usuario normal también puede incluir columnas (definidas con el tipo de datos DATALINK) que registren enlaces con datos almacenados en archivos externos. Los valores de DATALINK hacen referencia a archivos de datos almacenados en un servidor de archivos externo.

Conceptos relacionados:

- "Particionamiento de datos entre múltiples particiones" en la página 29

Información relacionada:

- "Sentencia CREATE BUFFERPOOL" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia CREATE DATABASE PARTITION GROUP" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia CREATE TABLESPACE" en la publicación *Consulta de SQL, Volumen 2*

Particionamiento de datos entre múltiples particiones

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

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

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

Antes de que se puedan utilizar las particiones para almacenar datos de la base de datos, deben definirse en el gestor de bases de datos. Las particiones se definen en un archivo llamado db2nodes.cfg.

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

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

1. Un algoritmo de cálculo de claves (función de particionamiento) se aplica a todas las columnas de la clave de particionamiento, lo que da como resultado la generación de un valor de índice de mapa de particionamiento.
2. El número de partición correspondiente a ese valor de índice en el mapa de particionamiento identifica la partición en la que se va a almacenar la fila.

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

DB2 tiene la posibilidad de reconocer si los datos a los que se está accediendo para una unión o subconsulta están situados en la misma partición del mismo grupo de particiones de base de datos. Esto se conoce como *colocación de tablas*. Las filas de las tablas colocadas con los mismos valores de clave de particionamiento se sitúan en la misma partición. DB2 puede elegir realizar el proceso de unión o subconsulta en la partición en la que están almacenados los datos. Esto puede tener ventajas significativas para el rendimiento.

Las tablas colocadas deben:

- Estar en el mismo grupo de particiones de base de datos, uno que no se esté redistribuyendo. (Durante la redistribución, es posible que las tablas del grupo de particiones de base de datos utilicen mapas de particionamiento distintos, que no estén colocadas).

- Tener claves de particionamiento con el mismo número de columnas.
- Hacer que las columnas correspondientes de la clave de particionamiento sean compatibles con la partición.
- Estar en un grupo de particiones de base de datos de una sola partición definido en la misma partición.

Información relacionada:

- "Tipos de datos compatibles entre particiones" en la página 131

Bases de datos relacionales distribuidas

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

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

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

Hay dos tipos de sentencias CONNECT:

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

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

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

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

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

Unidad de trabajo remota

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

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

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

En un momento dado, un proceso de aplicación se encuentra en uno de cuatro *estados de conexión* posibles:

- Conectable y conectado.

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

  Si la conexión implícita se encuentra disponible:

  – El proceso de aplicación entra en este estado cuando una sentencia CONNECT TO o una sentencia CONNECT sin operandos se ejecuta satisfactoriamente desde el estado conectable y no conectado.

  – El proceso de aplicación puede entrar en este estado desde el estado conectable implícitamente si se emite cualquier otra sentencia de SQL que no sea CONNECT RESET, DISCONNECT, SET CONNECTION ni RELEASE.

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

  – Se ejecuta satisfactoriamente una sentencia CONNECT TO desde el estado conectable y no conectado.

  – Se emite satisfactoriamente una sentencia COMMIT o ROLLBACK o tiene lugar una retrotracción forzada que procede de un estado no conectable y conectado.

- No conectable y conectado.

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

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

  Tanto si la conexión implícita está disponible como si no, el proceso de aplicación entra en este estado si se produce un error al emitir una sentencia CONNECT TO o si se produce un error en una unidad de trabajo que provoca la pérdida de la conexión y una retrotracción. Un error originado porque el proceso de aplicación no está en estado conectable o porque el nombre de servidor no se encuentra en el directorio local no provoca una transición a este estado.

  Si la conexión implícita no está disponible:
  - El proceso de aplicación está inicialmente en este estado
  - Las sentencias CONNECT RESET y DISCONNECT provocan una transición a este estado.
- Conectable implícitamente (si la conexión implícita se encuentra disponible).

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

La disponibilidad de la conexión implícita viene determinada por las opciones de la instalación, las variables de entorno y los valores de autenticación.

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

Unidad de trabajo remota

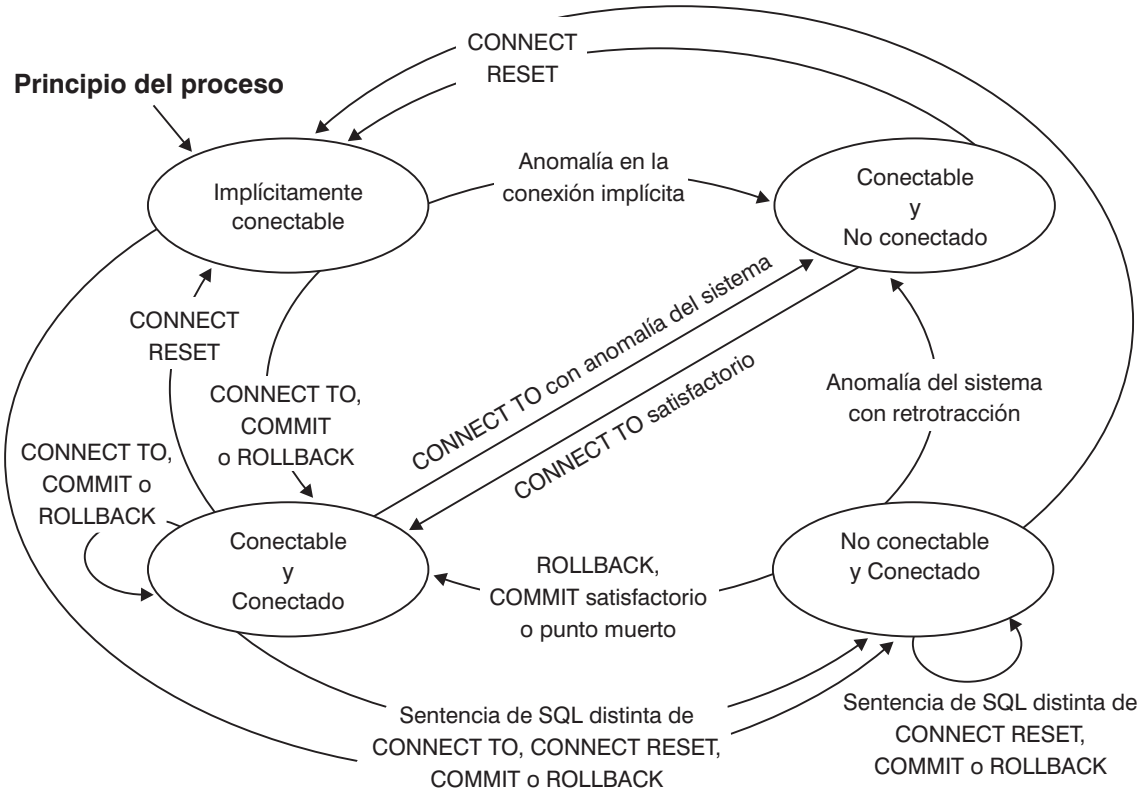


Figura 6. Transiciones de estado de conexión si la conexión implícita está disponible

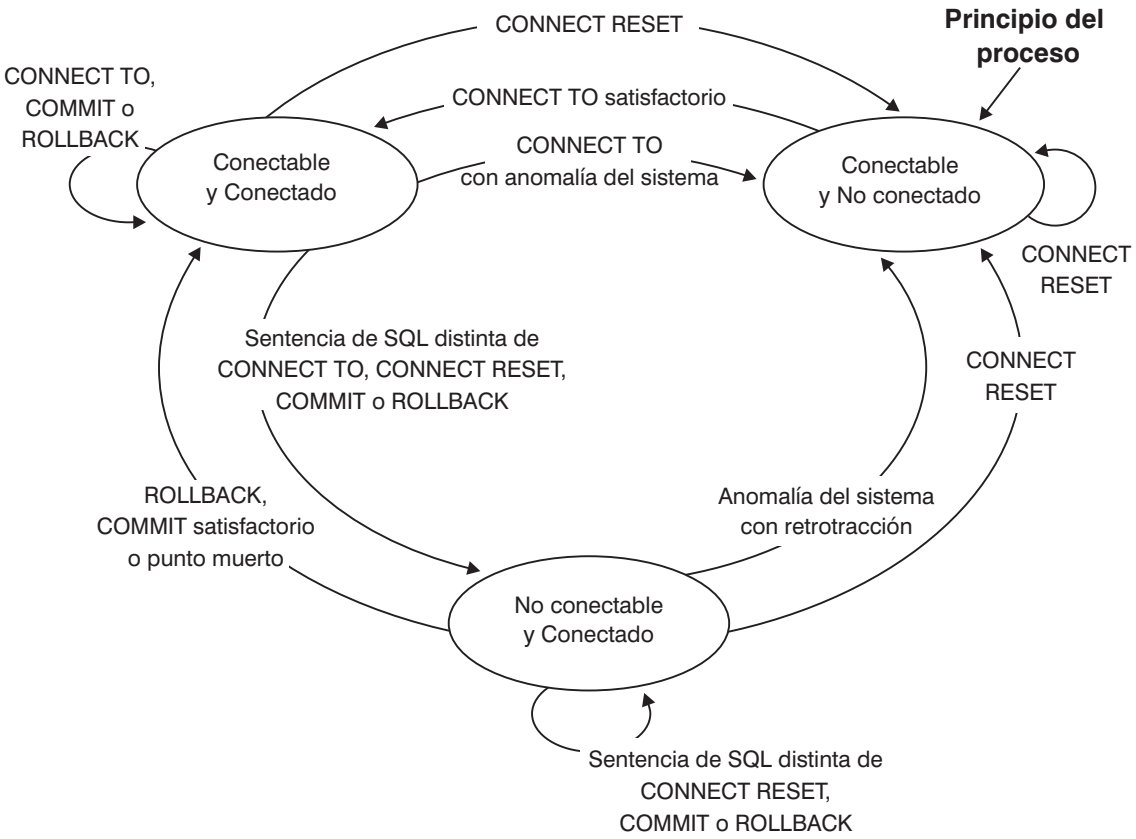


Figura 7. Transiciones de estado de conexión si la conexión implícita no está disponible

Unidad de trabajo distribuida dirigida por aplicación

El *recurso de unidad de trabajo distribuida dirigida por aplicación* también proporciona la preparación y ejecución remota de sentencias de SQL. Un proceso de aplicación en el sistema A puede conectarse a un servidor de aplicaciones en el sistema B emitiendo una sentencia CONNECT o SET CONNECTION. Después, el proceso de aplicación puede ejecutar un número cualquiera de sentencias de SQL estático y dinámico que hagan referencia a objetos de B antes de finalizar la unidad de trabajo. Todos los objetos a los que se hace referencia en una sola sentencia de SQL deben ser gestionados por el mismo servidor de aplicaciones. Sin embargo, a diferencia del recurso de unidad de trabajo remota, pueden participar en la misma unidad de trabajo cualquier número de servidores de aplicaciones. Una operación de retrotracción o confirmación finaliza la unidad de trabajo.

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

Un proceso de aplicación de tipo 2:

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

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

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

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

Un proceso de aplicación de tipo 2 está inicialmente en estado no conectado y no tiene ninguna conexión. Una conexión está inicialmente en estado actual y mantenido.

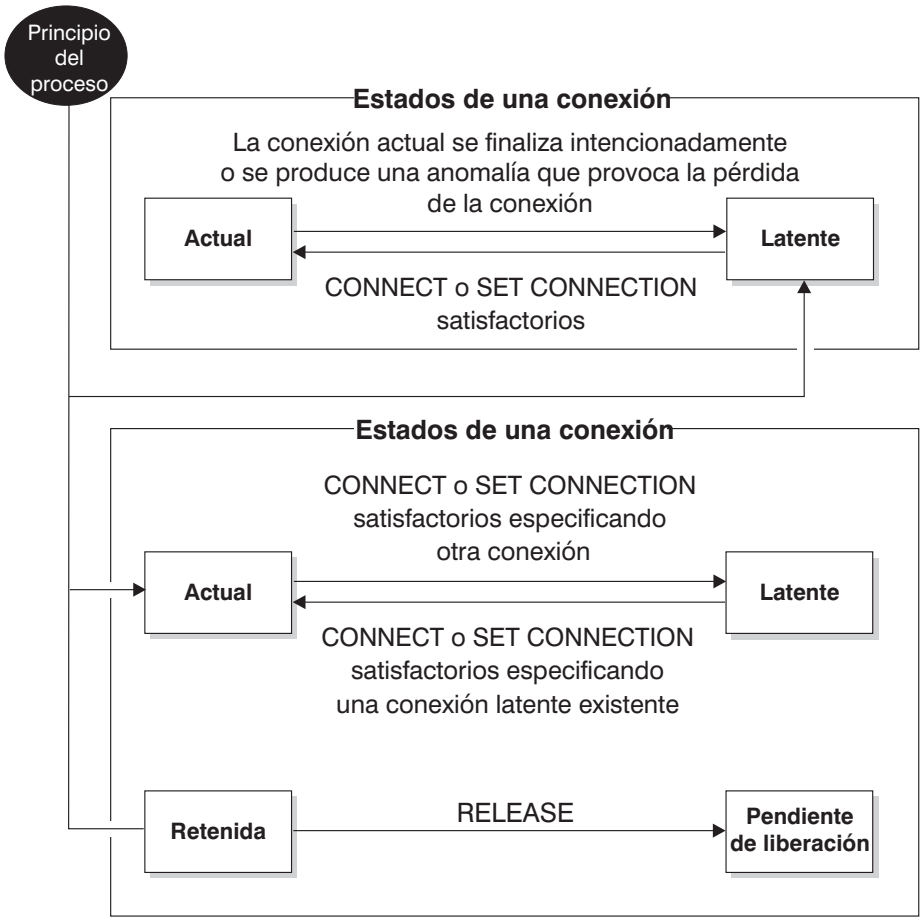


Figura 8. Transacciones de estado de conexión de unidad de trabajo distribuida dirigida por aplicación

Estados de conexión de procesos de aplicación

Se aplican las reglas siguientes a la ejecución de una sentencia CONNECT:

- Un contexto no puede tener más de una conexión al mismo servidor de aplicaciones al mismo tiempo.
- Cuando un proceso de aplicación ejecuta una sentencia SET CONNECTION, el nombre de ubicación especificado debe ser una conexión existente en el conjunto de conexiones del proceso de aplicación.
- Cuando un proceso de aplicación ejecuta una sentencia CONNECT y la opción SQLRULES(STD) está en vigor, el nombre de servidor especificado *no* debe ser una conexión existente en el conjunto de conexiones del proceso de aplicación.

Para ver una descripción de la opción SQLRULES, consulte el apartado "Opciones que controlan la semántica de la unidad de trabajo distribuida" en la página 38.

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

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

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

Un proceso de aplicación en el *estado conectado* pasa al *estado no conectado* cuando finaliza de manera intencionada su conexión actual o cuando una sentencia de SQL no es satisfactoria y provoca una operación de retrotracción en el servidor de aplicaciones y la pérdida de la conexión. Las conexiones finalizan intencionadamente al ejecutarse satisfactoriamente una sentencia DISCONNECT o una sentencia COMMIT cuando la conexión está en estado pendiente de liberación. (Si la opción DISCONNECT del precompilador está establecida en AUTOMATIC, finalizan todas las conexiones. Si está establecida en CONDITIONAL, finalizan todas las conexiones que no tienen cursores WITH HOLD abiertos.)

Estados de conexión

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

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

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

Estados mantenido y pendiente de liberación. La sentencia RELEASE controla si una conexión está en el estado mantenido o pendiente de liberación. El estado *pendiente de liberación* significa que se producirá una desconexión en la próxima operación de confirmación satisfactoria. (Una operación de retrotracción no tiene ningún efecto sobre las conexiones). El estado *mantenido* significa que *no* se producirá una desconexión en la próxima operación de confirmación.

Todas las conexiones se encuentran inicialmente en el estado mantenido y pueden pasar al estado pendiente de liberación utilizando la sentencia RELEASE. Una vez en el estado pendiente de liberación, una conexión no puede volver a pasar al

estado mantenido. Una conexión permanece en el estado pendiente de liberación más allá de los límites de la unidad de trabajo si se emite una sentencia ROLLBACK o si una operación de confirmación no satisfactoria da como resultado una operación de retrotracción.

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

Estados actual e inactivo. Con independencia de si una conexión se encuentra en el estado mantenido o pendiente de liberación, también puede estar en el estado actual o inactivo. Una conexión en el estado *actual* es la conexión que se está utilizando para ejecutar sentencias de SQL mientras se encuentra en este estado. Una conexión en el estado *inactivo* es una conexión que no es actual.

Las únicas sentencias de SQL que pueden fluir en una conexión inactiva son COMMIT, ROLLBACK, DISCONNECT y RELEASE. Las sentencias SET CONNECTION y CONNECT cambian el estado de conexión del servidor especificado a actual y cualquier conexión existente se coloca o permanece en estado inactivo. En todo momento, sólo una conexión puede estar en estado actual. Si una conexión inactiva pasa a ser actual en la misma unidad de trabajo, el estado de todos los bloqueos, cursores y sentencias preparadas es el mismo que el estado en estaban la última vez que la conexión era actual.

Cuando finaliza una conexión

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

Una conexión también puede finalizar a causa de una anomalía en las comunicaciones. Si esta conexión está en estado actual, el proceso de aplicación se coloca en estado no conectado.

Todas las conexiones de una proceso de aplicación finalizan cuando finaliza el proceso.

Opciones que controlan la semántica de la unidad de trabajo distribuida

La semántica de gestión de la conexión de tipo 2 viene determinada por un conjunto de opciones del precompilador. Estas opciones se resumen a continuación, con los valores por omisión indicados con texto en negrita y subrayado.

- CONNECT (**1** | 2). Especifica si las sentencias CONNECT se procesarán como tipo 1 o como tipo 2.
- SQLRULES (**DB2** | STD). Especifica si las sentencias CONNECT de tipo 2 deben procesarse según las reglas de DB2, que permiten que CONNECT conmute a una conexión inactiva, o según las reglas SQL92 Standard, que no permiten esta posibilidad.
- DISCONNECT (**EXPLICIT** | CONDITIONAL | AUTOMATIC). Especifica las conexiones de la base de datos que se desconectarán cuando se produzca una operación de confirmación:
  - Las que se han marcado explícitamente para su liberación mediante la sentencia de SQL RELEASE (EXPLICIT)

  - Las que no tienen cursores WITH HOLD abiertos y las que se han marcado para su liberación (CONDITIONAL)
  - Todas las conexiones (AUTOMATIC).
- SYNCPOINT (**ONEPHASE** | TWOPHASE | NONE). Especifica el modo en que se van a coordinar las operaciones COMMIT y ROLLBACK entre varias conexiones de bases de datos:
  - Las actualizaciones sólo pueden producirse respecto a una base de datos de la unidad de trabajo y el resto de las bases de datos son de sólo lectura (ONEPHASE). Cualquier intento de actualización en otra base de datos producirá un error (SQLSTATE 25000).
  - Se utiliza un gestor de transacciones (TM) en tiempo de ejecución para coordinar las operaciones COMMIT de dos fases entre las bases de datos que den soporte a este protocolo (TWOPHASE).
  - No utiliza ningún TM para realizar las operaciones COMMIT de dos fases y no impone un actualizador único y un lector múltiple (NONE). Cuando se ejecuta una sentencia COMMIT o ROLLBACK, las sentencias COMMIT o ROLLBACK individuales se envían a todas las bases de datos. Si hay una o varias operaciones ROLLBACK anómalas, se produce un error (SQLSTATE 58005). Si hay una o varias operaciones COMMIT anómalas, se produce un error (SQLSTATE 40003).

Para alterar temporalmente cualquiera de las opciones anteriores en tiempo de ejecución, utilice el mandato SET CLIENT o la interfaz de programación de aplicaciones (API) sqlesetc. Sus valores actuales se pueden obtener utilizando el mandato QUERY CLIENT o la API sqleqryc. Observe que no se trata de sentencias de SQL, sino que son API definidas en los distintos lenguajes principales y en el procesador de línea de mandatos (CLP).

Consideraciones sobre la representación de los datos

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

Información relacionada:

- "Sentencia CONNECT (Tipo 1)" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia CONNECT (Tipo 2)" en la publicación *Consulta de SQL, Volumen 2*

Sistemas federados de DB2

Sistemas federados

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

base de datos federada, una o varias fuentes de datos y clientes (usuarios y aplicaciones) que acceden a la base de datos y a las fuentes de datos. Con un sistema federado, puede enviar varias peticiones distribuidas a múltiples fuentes de datos dentro de una sentencia SQL individual. Por ejemplo, pueden unirse en una sola sentencia SQL datos ubicados en una tabla DB2 Universal Database™, una tabla Oracle y un archivo codificado en XML. La figura siguiente muestra los componentes de un sistema federado y una muestra de las fuentes de datos a las que es posible acceder.

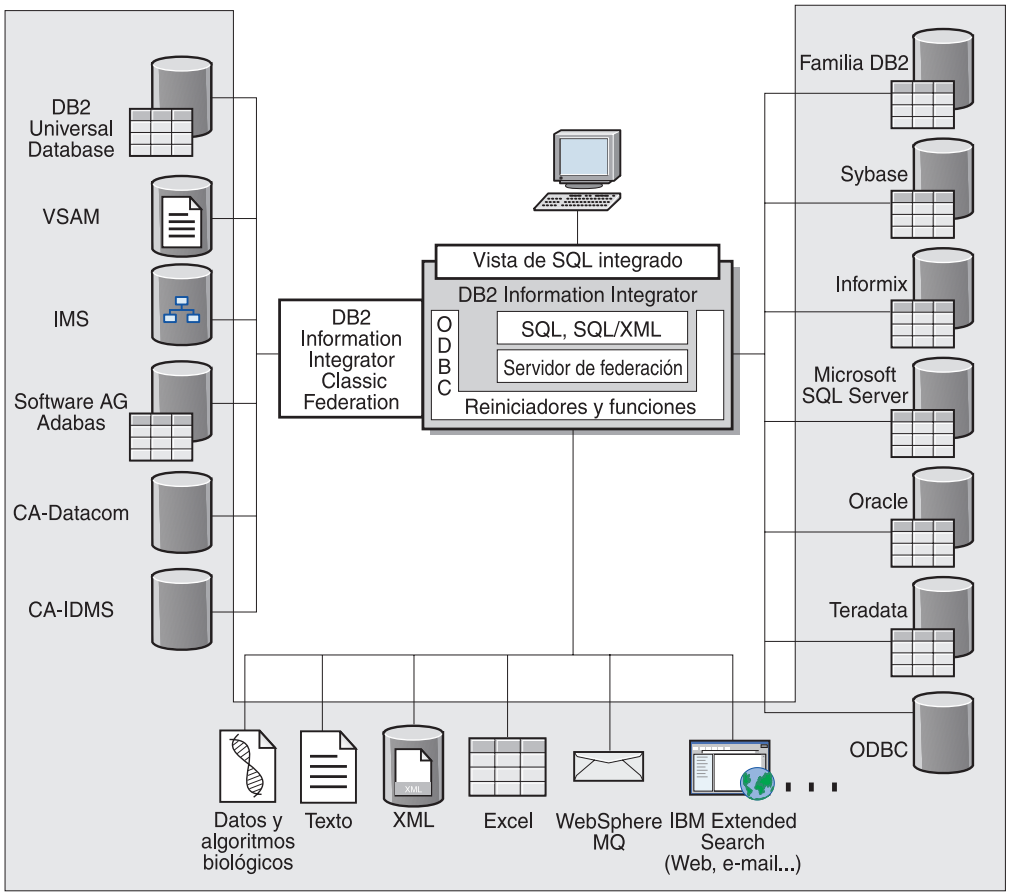


Figura 9. Los componentes de un sistema federado

El poder de un sistema federado de DB2 consiste en sus posibilidades de:

- Unir datos de tablas locales y fuentes de datos remotas como si todos los datos estuvieran almacenados localmente en la base de datos federada
- Actualizar los datos de fuentes de datos relacionales como si los datos estuvieran almacenados en la base de datos federada
- Duplicar los datos a partir de fuentes de datos relacionales y hacia las mismas
- Beneficiarse de las ventajas de procesar fuentes de datos enviando peticiones a las fuentes de datos para su proceso
- Compensar las limitaciones de SQL en la fuente de datos mediante el proceso de partes de una petición distribuida en el servidor federado

El servidor federado

El servidor DB2® de un sistema federado recibe el nombre de *servidor federado*. Puede configurarse cualquier número de instancias de DB2 para que funcionen como servidores federados. Es posible utilizar instancias de DB2 existentes como servidores federados o crear instancias nuevas de forma específica para el sistema federado.

La instancia de DB2 que gestiona el sistema federado se denomina un *servidor* porque responde a las peticiones de usuarios finales y aplicaciones clientes. El servidor federado a menudo envía partes de las peticiones que recibe a las fuentes de datos para su proceso. Una operación *de bajada* es una operación que se procesa de forma remota. La instancia que gestiona el sistema federado se denomina el *servidor federado* aunque actúe como cliente al bajar las peticiones a las fuentes de datos.

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

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

Conceptos relacionados:

- "¿Qué es una fuente de datos?" en la página 41

¿Qué es una fuente de datos?

En un sistema federado, una *fuente de datos* puede ser una instancia de DBMS relacional (como, por ejemplo, Oracle o Sybase) o una fuente de datos no relacional (como, por ejemplo, el algoritmo de búsqueda de BLAST o un archivo codificado en XML). Mediante algunas fuentes de datos es posible acceder a otras fuentes de datos. Por ejemplo, mediante la fuente de datos Extended Search es posible acceder a fuentes de datos como las bases de datos de Lotus® Notes, Microsoft® Access, Microsoft Index Server, los motores de búsqueda Web y los directorios del protocolo LDAP.

El método o protocolo utilizado para acceder a la fuente de datos depende del tipo de fuente de datos. Por ejemplo, se utiliza DRDA® para acceder a fuentes de datos DB2® para z/OS™ y OS/390® y se utiliza la API/biblioteca del cliente de Documentum para acceder a fuentes de datos Documentum.

Las fuentes de datos son semiautónomas. Por ejemplo, el servidor federado puede enviar consultas a fuentes de datos Oracle al mismo tiempo que aplicaciones

Oracle acceden a estas fuentes de datos. Un sistema federado DB2 no monopoliza ni restringe el acceso a otras fuentes de datos más allá de las restricciones de integridad y de bloqueo.

Conceptos relacionados:

- "La base de datos federada" en la página 44

Información relacionada:

- "Fuentes de datos soportadas" en la página 42

Fuentes de datos soportadas

Existen muchas fuentes de datos a las que es posible acceder utilizando un sistema federado. La tabla siguiente muestra las fuentes de datos soportadas:

Tabla 1. Versiones de fuentes de datos soportadas y métodos de acceso.

Fuente de datos	Versiones soportadas	Método de acceso
DB2 Universal Database™ para Linux, UNIX y Windows®	7.2, 8.1, 8.2	DRDA
DB2 Universal Database para z/OS y OS/390	6.1, 7.1 con las siguientes APAR aplicadas: • PQ62695 • PQ55393 • PQ56616 • PQ54605 • PQ46183 • PQ62139 8.1	DRDA
DB2 Universal Database para iSeries	5.1 • con las siguientes APAR aplicadas: – SE06003 – SE06872 – II13348 • con los siguientes PTF aplicados: – SI05990 SI05991 5.2 con el PTF SI0735 aplicado.	DRDA
DB2 Server para VM y VSE	7.1 (o posterior) con los arreglos para las APAR para funciones de esquema aplicados.	DRDA
Informix	7.31, 8.32, 8.4, 9.3, 9.4	SDK de Cliente Informix V 2.7 (o posterior)

Tabla 1. Versiones de fuentes de datos soportadas y métodos de acceso. (continuación)

Fuente de datos	Versiones soportadas	Método de acceso
ODBC	3.x	Controlador ODBC para la fuente de datos como, por ejemplo, el controlador ODBC de Redbrick para acceder a Redbrick.
OLE DB	2.7, 2.8	OLE DB 2.0 (o posterior)
Oracle	8.0.6, 8.1.6, 8.1.7, 9.0, 9.1, 9.2, 9i, 10g	Software de cliente de red Oracle o cliente NET8
Microsoft SQL Server	7.0, 2000 SP3 y service packs posteriores de este release	En Windows, el controlador ODBC 3.0 (o posterior) del cliente Microsoft SQL Server. En UNIX, el controlador Connect ODBC 3.7 de DataDirect Technologies (anteriormente MERANT).
Sybase	11.9.2, 12.x	Interfaz CTLIB de Sybase Open Client
Teradata	V2R3, V2R4, V2R5	Interfaz de nivel de llamada de Teradata Versión 2 (CLIv2) Release 04.06 (o posterior)
BLAST	2.2.3 y fixpacks soportados posteriores 2.2	Daemon BLAST (suministrado con el reiniciador)
BioRS	v5.0.14	Ninguno
Documentum	3.x, 4.x	Biblioteca del cliente de Documentum/APL3.1.7a (o posterior)
Entrez (fuentes de datos PubMed y GenBank)	1.0	Ninguno
HMMER	2.2g, 2.3	Daemon HMMER (suministrado con el reiniciador)
IBM Lotus Extended Search	4.0.1, 4.0.2	Biblioteca de cliente de Extended Search(suministrada con el reiniciador)
Microsoft Excel	97, 2000, 2002, 2003	Excel 97, 2000, 2002 o 2003 instalado en el servidor federado
PeopleSoft	8.x	Adaptador de IBM Integración comercial de WebSphere para PeopleSoft v2.3.1, 2.4
SAP	3.x, 4.x	Adaptador de IBM Integración comercial de WebSphere para mySAP.com v2.3.1, 2.4

Tabla 1. Versiones de fuentes de datos soportadas y métodos de acceso. (continuación)

Fuente de datos	Versiones soportadas	Método de acceso
Siebel	7, 7.5, 2000	Adaptador de IBM Integración comercial de WebSphere para Siebel eBusiness Applications v2.3.1, 2.4
Archivos estructurados-tabla		Ninguno
Funciones definidas por el usuario para KEGG	Recibe soporte	
Funciones biológicas definidas por el usuario	Soportado	
Servicios Web	Especificaciones SOAP 1.0, 1.1, WSDL 1.0, 1.1	HTTP
XML	Especificación 1.0	Ninguno

Conceptos relacionados:

• "¿Qué es una fuente de datos?" en la página 41

La base de datos federada

Para los usuarios finales y aplicaciones cliente, las fuentes de datos aparecen como una sola base de datos colectiva en DB2®. Los usuarios y las aplicaciones interactúan con la *base de datos federada* gestionada por el servidor federado. La base de datos federada contiene un catálogo del sistema. El catálogo del sistema de la base de datos federada contiene entradas que identifican las fuentes de datos y las características de las mismas. El servidor federado consulta la información almacenada en catálogo del sistema de bases de datos federadas y el reiniciador de fuentes de datos para determinar el mejor plan para procesar las sentencias de SQL.

El sistema federado procesa las sentencias de SQL como si las fuentes de datos fueran tablas relacionales corrientes dentro de la base de datos federada. Como resultado:

- El sistema federado puede unir datos relacionales con datos en formatos no relacionales. Esto es así incluso cuando las fuentes de datos utilizan dialectos SQL diferentes o ni siquiera proporcionan soporte al SQL.
- Las características de la base de datos federada tienen prioridad en caso de diferencias entre las características de la base de datos federada y las características de las fuentes de datos:
  - Supongamos que la página de códigos utilizada por el servidor federado es distinta de la página de códigos utilizada por la fuente de datos. Los datos de caracteres de la fuente de datos se convierten de acuerdo a la página de códigos que la base de datos federada utiliza cuando estos datos se devuelven a un usuario federado.
  - Supongamos que el orden de clasificación utilizado por el servidor federado es distinto del orden de clasificación utilizado por la fuente de datos. Las operaciones de clasificación efectuadas sobre datos de caracteres se realizarán en el servidor federado en lugar de en la fuente de datos.

Conceptos relacionados:

- "El compilador de SQL" en la página 46
- "El catálogo del sistema de bases de datos federadas" en la página 45

El catálogo del sistema de bases de datos federadas

El catálogo del sistema de bases de datos federadas contiene información sobre los objetos de la base de datos federada e información sobre los objetos de las fuentes de datos. El catálogo de una base de datos federada se denomina el *catálogo global* porque contiene información sobre todo el sistema federado. El optimizador de consultas de DB2 utiliza la información del catálogo global y el reiniciador de fuentes de datos para planificar la mejor forma de procesar las sentencias de SQL. La información almacenada en el catálogo global incluye información remota y local como, por ejemplo, los nombres de las columnas, los tipos de datos de las columnas, los valores por omisión de las columnas y la información sobre el índice.

La información de catálogo *remota* es la información o el nombre que la fuente de datos utiliza. La información de catálogo *local* es la información o el nombre que la base de datos federada utiliza. Por ejemplo, supongamos que una tabla remota incluye una columna con el nombre de *NUMEMP*. El catálogo global almacenaría el nombre de la columna remota como *NUMEMP*. A menos que designe un nombre distinto, el nombre de columna local se almacenará como *NUMEMP*. Es posible cambiar el nombre de la columna local a *Número_Empleado*. Los usuarios que sometan consultas que incluyan este columna utilizarán *Número_Empleado* en sus consultas en lugar de *NUMEMP*. La sentencia ALTER NICKNAME se utiliza para cambiar el nombre local de las columnas de la fuente de datos.

Para fuentes de datos relacionales, la información almacenada en el catálogo global incluye información tanto remota como local.

Para fuentes de datos no relacionales, la información almacenada en el catálogo global varía de una fuente de datos a otra.

Para ver la información sobre las tablas de la fuente de datos que está almacenada en el catálogo global, consulte las vistas de catálogo SYSCAT.TABLES, SYSCAT.TABOPTIONS, SYSCAT.INDEXES, SYSCAT.COLUMNS y SYSCAT.COLOPTIONS.

El catálogo global también incluye otra información acerca de las fuentes de datos. Por ejemplo, el catálogo global incluye información que el servidor federado utiliza para conectarse a la fuente de datos y correlacionar las autorizaciones de los usuarios federados con las autorizaciones de los usuarios de la fuente de datos. El catálogo global contiene atributos acerca de la fuente de datos que deben definirse de forma explícita como, por ejemplo, las opciones del servidor.

Conceptos relacionados:

- "El compilador de SQL" en la página 46

Información relacionada:

- "Views in the global catalog table containing federated information" en la publicación *Federated Systems Guide*

El compilador de SQL

Para obtener datos de las fuentes de datos, los usuarios y las aplicaciones someten consultas en SQL de DB2 a la base de datos federada. Cuando se somete una consulta, el compilador de SQL de DB2 consulta la información en el catálogo global y el reiniciador de fuentes de datos para ayudarlo a procesar la consulta. Esto incluye información sobre cómo conectarse a la fuente de datos, atributos de servidor, correlaciones, información sobre los índices y datos estadísticos de procesos.

Conceptos relacionados:

- "Reiniciadores y módulos de reiniciador" en la página 49
- "El optimizador de consultas" en la página 46

El optimizador de consultas

Como parte del proceso del compilador de SQL, el *optimizador de consultas* analiza una consulta. El Compilador desarrolla estrategias alternativas, denominadas *planes de acceso*, para procesar la consulta. Los planes de acceso podrían llamar a la consulta para que:

- La procesaran las fuentes de datos
- La procesa el servidor federado
- La procesaran en parte las fuentes de datos y en parte el servidor federado

DB2® UDB evalúa los planes de acceso principalmente en base a la información sobre las posibilidades de las fuentes de datos y los datos. El reiniciador y el catálogo global contienen esta información. DB2 UDB descompone la consulta en segmentos que se denominan *fragmentos de consulta*. Normalmente es más efectivo *bajar* un fragmento de consulta a la fuente de datos si la fuente de datos puede procesar el fragmento. Sin embargo, el optimizador de consultas tiene en cuenta otras funciones como, por ejemplo:

- La cantidad de datos que debe procesarse
- La velocidad de proceso de la fuente de datos
- La cantidad de datos que devolverá el fragmento
- El ancho de banda de la comunicación
- Si existe una tabla de consulta materializada en el servidor federado que represente el mismo resultado de la consulta

El optimizador de consultas genera planes de acceso locales y remotos para procesar un fragmento de consulta, en función del coste de recursos. DB2 UDB elige entonces el plan que cree que procesará la consulta con el mínimo coste de recursos.

Si las fuentes de datos deben procesar alguno de los fragmentos, DB2 UDB somete estos fragmentos a las fuentes de datos. Después de que las fuentes de datos procesen los fragmentos, los resultados se recuperan y se devuelven a DB2 UDB. Si DB2 UDB ha realizado alguna parte del proceso, combina sus resultados con los resultados recuperados de la fuente de datos. DB2 UDB devuelve entonces todos los resultados al cliente.

Conceptos relacionados:

- "El compilador de SQL" en la página 46

- "Compensación" en la página 47
- "Tuning query processing" en la publicación *Federated Systems Guide*

Compensación

El servidor federado de DB2 no baja un fragmento de consulta si la fuente de datos no puede procesarlo o si el servidor federado puede procesarlo con mayor rapidez que la fuente de datos. Por ejemplo, supongamos que el dialecto SQL de una fuente de datos no proporcione soporte a una agrupación CUBE en la cláusula GROUP BY. Una consulta que contenga la agrupación CUBE y haga referencia a una tabla de la fuente de datos se envía al servidor federado. DB2 Information Integrator no baja la agrupación CUBE a la fuente de datos sino que procesa la agrupación CUBE directamente. La posibilidad que posee DB2 Information Integrator de procesar el SQL al que la fuente de datos no proporcione soporte se denomina *compensación*.

El servidor federado compensa la falta de funcionalidad de la fuente de datos de dos formas:

- Puede pedir a la fuente de datos que utilice una o varias operaciones que sean equivalentes a la función de DB2 utilizada en la consulta. Supongamos que una fuente de datos no proporcione soporte a la función de cotangente (COT(x)) pero sí a la función de tangente (TAN(x)). DB2 Information Integrator puede pedir a la fuente de datos que realice el cálculo (1/TAN(x)), que es equivalente a la función de cotangente (COT(x)).
- Puede devolver el conjunto de datos al servidor federado y realizar la función localmente.

Para las fuentes de datos relacionales, cada tipo de RDBMS proporciona soporte a un subconjunto del estándar internacional de SQL. Además, algunos tipos de RDBMS proporcionan soporte a construcciones SQL que exceden este estándar. Un *dialecto SQL* es la totalidad de SQL a la que un tipo de RDBMS proporciona soporte. Si una construcción SQL se encuentra en el dialecto SQL de DB2 pero no en el dialecto de la fuente de datos relacional, el servidor federado puede implementar esta construcción en nombre de la fuente de datos.

DB2 Information Integrator puede compensar las diferencias entre dialectos de SQL. Un ejemplo de esta posibilidad se observa en la cláusula expresión-tabla-común. El SQL de DB2 incluye la cláusula expresión-tabla-común. En esta cláusula, se puede especificar un nombre mediante el cual todas las cláusulas FROM de una selección completa pueden hacer referencia a un conjunto resultante. El servidor federado procesará una expresión-tabla-común para fuente de datos aunque el dialecto SQL que la fuente de datos utiliza no incluya la expresión-tabla-común.

Mediante la compensación, el servidor federado puede proporcionar soporte a la totalidad del dialecto SQL de DB2 para realizar consultas de las fuentes de datos. Incluso las fuentes de datos con un soporte al SQL escaso o que no proporcionen soporte a SQL se beneficiarán de la compensación. El dialecto SQL de DB2 debe utilizarse con un sistema federado, a excepción de las sesiones de paso a través.

Conceptos relacionados:

- "Sesiones de paso a través" en la página 48

Sesiones de paso a través

Es posible enviar sentencias de SQL directamente a las fuentes de datos utilizando una modalidad especial llamada *paso a través*. Las sentencias de SQL se envían en el dialecto SQL que la fuente de datos utiliza. Utilice una sesión de paso a través cuando desee realizar una operación que no sea posible con SQL/API de DB2. Por ejemplo, utilice una sesión de paso a través para crear un procedimiento, crear un índice o realizar consultas en el dialecto nativo de la fuente de datos.

Actualmente, las fuentes de datos que proporcionan soporte al paso a través proporcionan el paso a través utilizando SQL. En el futuro, es posible que las fuentes de datos proporcionen soporte al paso a través utilizando una fuente de datos distinta de SQL.

De forma similar, es posible utilizar una sesión de paso a través para realizar acciones a las que SQL no proporcione soporte como, por ejemplo, ciertas tareas administrativas. Sin embargo, no es posible utilizar una sesión de paso a través para realizar todas las tareas administrativas. Por ejemplo, es posible crear y eliminar tablas en la fuente de datos pero no iniciar o detener la base de datos remota.

En una sesión de paso a través puede utilizarse SQL estático o dinámico.

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

SET PASSTHRU
: Abre una sesión de paso a través. Cuando se emite otra sentencia SET PASSTHRU para iniciar una nueva sesión de paso a través, se termina la sesión de paso a través actual.

SET PASSTHRU RESET
: Termina la sesión de paso a través actual.

GRANT (Privilegios de servidor)
: Otorga a un usuario, grupo, lista de ID de autorización o PUBLIC la autorización para iniciar sesiones de paso a través para una fuente de datos específica.

REVOKE (Privilegios del servidor)
: Revoca la autorización para iniciar sesiones de paso a través.

Las restricciones siguientes son aplicables a las sesiones de paso a través:

- Debe utilizarse el dialecto SQL o los mandatos de lenguaje de la fuente de datos— no es posible utilizar el dialecto SQL de DB2. Por lo tanto, las consultas no se realizan en apodos, sino directamente en los objetos de la fuente de datos.
- Al realizar una operación UPDATE o DELETE en una sesión de paso a través, no es posible utilizar la condición WHERE CURRENT OF CURSOR.
- En las sesiones de paso a través no se proporciona soporte a los LOB.

Conceptos relacionados:

- "Reiniciadores y módulos de reiniciador" en la página 49
- "Querying data sources directly with pass-through" en la publicación *Federated Systems Guide*

Reiniciadores y módulos de reiniciador

Los *reiniciadores* son mecanismos mediante los cuales el servidor federado interactúa con las fuentes de datos. El servidor federado utiliza las rutinas almacenadas en una biblioteca denominada *módulo de reiniciador* para implantar un reiniciador. Estas rutinas permiten al servidor federado realizar operaciones como, por ejemplo, conectarse a una fuente de datos y recuperar datos de la misma repetidamente. Normalmente, el propietario de la instancia federada de DB2® utiliza la sentencia CREATE WRAPPER para registrar un reiniciador en la base de datos federada. Un reiniciador puede registrarse como delimitado o de confianza utilizando la opción del reiniciador DB2_FENCED.

Debe crearse un reiniciador para cada tipo de fuente de datos al que se desee acceder. Por ejemplo, supongamos que desea acceder a tres tablas de la base de datos DB2 para z/OS™, a una tabla DB2 para iSeries™, a dos tablas Informix® y a una vista Informix. Necesita crear un reiniciador para los objetos de la fuente de datos de DB2 y otro para los objetos de la fuente de datos de Informix. Cuando se hayan registrado estos reiniciadores en la base de datos federada, podrá utilizarlos para acceder a otros objetos de estas fuentes de datos. Por ejemplo, puede utilizar el reiniciador DRDA® con todos los objetos de la fuente de datos de la familia DB2: DB2 para Linux, UNIX® y Windows®, DB2 para z/OS y OS/390®, DB2 para iSeries y DB2 Server para VM y VSE.

Las definiciones y los apodos del servidor se utilizan para identificar los detalles (nombre, ubicación, etc.) de cada objeto de fuente de datos.

Un reiniciador realiza muchas tareas. Algunas de estas tareas son las siguientes:

- Conecta con la fuente de datos. El reiniciador utiliza la API de conexión estándar de la fuente de datos.
- Envía consultas a la fuente de datos.
  - Para las fuentes de datos que sí que proporcionan soporte a SQL, la consulta se envía en SQL.
  - Para las fuentes de datos que no proporcionan soporte a SQL, la consulta se convierte al lenguaje de consulta nativo de la fuente a a una serie de llamadas a la API de la fuente.
- Recibe el conjuntos de resultados de la fuente de datos. El reiniciador utiliza las API estándar de fuentes de datos para recibir el conjunto de resultados.
- Responde a las consultas del servidor federado sobre las correlaciones de tipos de datos por omisión para una fuente de datos. El reiniciador contiene las correlaciones de tipos por omisión que se utilizan cuando se crean apodos para un objeto de la fuente de datos. Para los reiniciadores relacionales, las correlaciones de tipos de datos que se crean alteran temporalmente las correlaciones de tipos de datos por omisión. Las correlaciones de tipos de datos definidos por el usuario se almacenan en el catálogo global.
- Responde a las consultas del servidor federado sobre las correlaciones de funciones por omisión para una fuente de datos. El reiniciador contiene información que el servidor federado necesita para determinar si las funciones de DB2 se correlacionan con funciones de la fuente de datos y de qué forma lo hacen. El compilador de SQL utiliza esta información para determinar si la fuente de datos puede realizar las operaciones de consulta. Para los reiniciadores relacionales, las correlaciones de funciones que se crean alteran temporalmente las correlaciones de tipos de funciones por omisión. Las correlaciones de funciones definidas por el usuario se almacenan en el catálogo global.

Las *opciones del reiniciador* se utilizan para configurar el reiniciador o definir la forma en que DB2 Information Integrator utiliza el reiniciador.

Conceptos relacionados:

- "Definiciones de servidor y opciones de servidor" en la página 51

Tareas relacionadas:

- "Entornos de proceso de modalidad protegida o fiable" en la publicación *IBM DB2 Information Integrator Wrapper Developer's Guide*
- "Altering a wrapper" en la publicación *Federated Systems Guide*

Información relacionada:

- "Sentencia ALTER WRAPPER" en la publicación *Consulta de SQL, Volumen 2*
- "Opciones del reiniciador para sistemas federados" en la página 699
- "Nombres de reiniciadores por omisión" en la página 50

Nombres de reiniciadores por omisión

Existen reiniciadores para cada fuente de datos soportada. Algunos reiniciadores tienen nombres de reiniciadores por omisión. Cuando utiliza el nombre por omisión para crear el reiniciador, el servidor federado reúne automáticamente la biblioteca de la fuente de datos asociada con el reiniciador.

Tabla 2. Nombres de reiniciador por omisión para cada fuente de datos.

Fuente de datos	Nombres de reiniciadores por omisión
DB2 Universal Database™ para Linux, UNIX y Windows®	DRDA
DB2 Universal Database para z/OS y OS/390	DRDA
DB2 Universal Database para iSeries	DRDA
DB2 Server para VM y VSE	DRDA
Informix	INFORMIX
Microsoft SQL Server	MSSQLODBC3
ODBC	ODBC
OLE DB	OLEDB
Oracle	NET8
Sybase	CTLIB
Teradata	TERADATA
BLAST	Ninguno
BioRS	Ninguno
Documentum	Ninguno
Entrez	Ninguno
Extended Search	Ninguno
HMMER	Ninguno
Microsoft Excel	Ninguno
Archivos estructurados-tabla	Ninguno
Servicios Web	Ninguno
Integración comercial de WebSphere	Ninguno

Tabla 2. Nombres de reiniciador por omisión para cada fuente de datos. (continuación)

Fuente de datos	Nombres de reiniciadores por omisión
XML	Ninguno

Conceptos relacionados:

- "Reiniciadores y módulos de reiniciador" en la página 49

Definiciones de servidor y opciones de servidor

Después de crear reiniciadores para las fuentes de datos, el propietario de la instancia federada debe definir las fuentes de datos a la base de datos federada. El propietario de la instancia suministra un nombre para identificar la fuente de datos y otra información que pertenece a la fuente de datos. Esta información incluye lo siguiente:

- El tipo y la versión de la fuente de datos
- El nombre de la base de datos de la fuente de datos (sólo para RDBMS)
- Los metadatos específicos para la fuente de datos

Por ejemplo, una fuente de datos de la familia DB2 puede tener múltiples bases de datos. La definición debe especificar a qué base de datos puede conectarse un servidor federado. Por el contrario, una fuente de datos Oracle tiene una base de datos y el servidor federado puede conectarse a la base de datos sin conocer su nombre. El nombre de la base de datos no se incluye en la definición de servidor federado de una fuente de datos Oracle.

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

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

Para de la información incluida en una definición de servidor se almacena como *opciones de servidor*. Al crear definiciones de servidor, es importante comprender las opciones que se pueden especificar sobre el servidor. Algunas opciones de servidor configuran el reiniciador y algunas afectan a la forma en que DB2 Information Integrator utiliza el reiniciador.

Las opciones del servidor pueden establecerse de forma que persistan durante conexiones sucesivas a la fuente de datos o pueden establecerse para que duren una sola conexión.

Conceptos relacionados:

- "Correlaciones de usuarios" en la página 52

Información relacionada:

- "Opciones del servidor para sistemas federados" en la página 683

Correlaciones de usuarios

Cuando un servidor federado necesita bajar una petición a una fuente de datos, el servidor debe establecer primero una conexión con la fuente de datos.

Para la mayoría de fuentes de datos, el servidor federado lleva a cabo esta operación utilizando un ID de usuario y una contraseña válidos para la fuente de datos. Cuando se necesita un ID de usuario y una contraseña para conectarse con la fuente de datos, es posible definir una asociación entre el ID de autorización del servidor federado y el ID de usuario de la fuente de datos y la contraseña. Esta asociación se puede crea para cada ID de usuario que el sistema federado vaya a utilizar para enviar peticiones distribuidas. Esta asociación se denomina *correlación de usuarios*.

En algunos casos, no es necesario crear una correlación de usuarios si el ID de usuario y la contraseña que utiliza para conectarse a la base de datos federada coinciden con los que utiliza para acceder a la fuente de datos remota.

Tareas relacionadas:

- "Cómo registrar correlaciones de usuarios para una fuente de datos" en la publicación *IBM DB2 Information Integrator Guía de configuración de fuentes de datos*

Apodos y objetos de la fuente de datos

Después de crear las definiciones del servidor y las correlaciones del usuario, el propietario de la instancia federada debe crear los apodos. Un *apodo* es un identificador que se utiliza para hacer referencia al objeto ubicado en las fuentes de datos al que desea acceder. Los objetos identificados mediante apodos se denominan *objetos de la fuente de datos*.

Los apodos no son nombres alternativos para los objetos de la fuente de datos del modo en que lo son los seudónimos. Son punteros mediante los cuales el servidor federado hace referencia a estos objetos. Los apodos suelen definirse con la sentencia CREATE NICKNAME junto con las opciones específicas de la columna de apodo y las opciones de apodos.

Cuando un usuario final o una aplicación cliente somete una petición distribuida al servidor federado, no es necesario que la petición especifique las fuentes de datos. En lugar de ello, la petición hace referencia a los objetos de la fuente de datos mediante sus apodos. Los apodos se correlacionan con los objetos concretos en la fuente de datos. Estas correlaciones eliminan la necesidad de calificar los apodos mediante los nombres de las fuentes de datos. La ubicación de los objetos de la fuente de datos es transparente para el usuario final o la aplicación cliente.

Supongamos que se define el apodo *DEPT* para representar una tabla de la base de datos de Informix® denominada *NFX1.PERSON*.Desde el servidor federado puede ejecutarse la sentencia SELECT * FROM *DEPT*. Sin embargo, la sentencia SELECT * FROM *NFX1.PERSON* no puede ejecutarse desde el servidor federado (a excepción de las sesiones de paso a través)a menos que exista una tabla local en el servidor federado con el nombre *NFX1.PERSON*.

Cuando crea un apodo para un objeto de fuente de datos, se añaden metadatos sobre el objeto al catálogo global. El optimizador de consultas utiliza estos metadatos y la información del reiniciador para facilitar el acceso al objeto de la

fuente de datos. Por ejemplo, si el apodo es para una tabla que tiene un índice, el catálogo global contiene información acerca del índice. El reiniciador contiene las correlaciones entre los tipos de datos de DB2® y los tipos de datos de la fuente de datos.

Actualmente, no es posible ejecutar algunas de las operaciones del programa de utilidad UDB de DB2 sobre apodos.

No es posible utilizar el programa de utilidad Cross Loader para realizar una carga cruzada en un apodo.

Conceptos relacionados:

- "Opciones de columna de apodo" en la página 54

Información relacionada:

- "Opciones de la columna de apodo para sistemas federados" en la página 675
- "Nickname options for federated systems" en la publicación *Federated Systems Guide*
- "Objetos válidos para fuentes de datos" en la página 53

Objetos válidos para fuentes de datos

Los apodos identifican los objetos de la fuente de datos a la que desea acceder. La tabla siguiente muestra los tipos de objetos para los que puede crearse un apodo en un sistema federado.

Tabla 3. Objetos válidos de fuentes de datos

Fuente de datos	Objetos válidos
DB2 para Linux, UNIX y Windows	Apodos, tablas de consulta materializada, tablas, vistas
DB2 para z/OS y OS/390	Tablas, vistas
DB2 para iSeries	Tablas, vistas
DB2 para VM y VSE	Tablas, vistas
Informix	tablas, vistas, sinónimos
Microsoft SQL Server	Tablas, vistas
ODBC	Tablas, vistas
Oracle	tablas, vistas, sinónimos
Sybase	Tablas, vistas
Teradata	Tablas, vistas
BLAST	archivos FASTA indexados para algoritmos de búsqueda BLAST
BioRS	Banco de datos BioRS
Documentum	Objetos y tablas registradas en una base de documentos de Documentum
Entrez	Bases de datos de Entrez
Extended Search	Archivos de fuentes de datos como, por ejemplo, bases de datos de Lotus Notes, Microsoft Access, Microsoft Index Server, los motores de búsqueda Web y los directorios LDAP.

Tabla 3. Objetos válidos de fuentes de datos (continuación)

Fuente de datos	Objetos válidos
HMMER	Archivos de bases de datos HMM (bibliotecas de modelos Markov jerárquicos como, por ejemplo, PFAM) en los que sea posible realizar búsquedas utilizando los programas hmmpfam o hmmsearch de HMMER.
Microsoft Excel	archivos .xls (sólo se accede a la primera hoja del libro de trabajo)
Archivos estructurados-tabla	Archivos de texto que cumplan un formato determinado.
Adaptadores de Integración comercial de Websphere	Objetos de Integración comercial de Websphere que se correlacionen con BAPI en SAP, componentes comerciales en Siebel e interfaces de componentes en PeopleSoft
Servicios Web	Operaciones en un archivo del lenguaje de descripción de servicios Web (WSDL).
Archivos codificados en XML	Conjuntos de elementos en un documento XML

Conceptos relacionados:

- "Apodos y objetos de la fuente de datos" en la página 52
- "Opciones de columna de apodo" en la página 54

Opciones de columna de apodo

Es posible proporcionar al catálogo global información adicional sobre el objeto al que el apodo hace referencia mediante metadatos. Estos metadatos describen valores en ciertas columnas del objeto de la fuente de datos. Estos metadatos se asignan a parámetros que se denominan *opciones de columna de apodo*. Las opciones de columna de apodo indican al reiniciador que gestione los datos de una columna de forma distinta a como lo haría normalmente. El compilador de SQL y optimizador de consultas utilizan los metadatos para desarrollar planes mejores para acceder a los datos.

Las opciones de columna de apodo también se utilizan para proporciona otra información al reiniciador. Por ejemplo, para las fuentes de datos XML, se utiliza una opción de columna de apodo para indicar al reiniciador qué expresión XPath debe utilizar al analizar la columna fuera del documento XML.

Con la federación, el servidor DB2® trata al objeto de fuente de datos al que un apodo hace referencia como si se tratase una tabla local de DB2. Por lo tanto, es posible definir opciones de columna de apodo para cualquier objeto de la fuente de datos para el que cree un apodo. Algunas opciones de columna de apodo están diseñadas para tipos concretos de fuentes de datos y sólo pueden aplicarse a estas fuentes de datos.

Supongamos que una fuente de datos tiene un orden de clasificación que difiere del orden de clasificación de la base de datos federada. El servidor federado normalmente no clasificaría las columnas que contienen datos de caracteres de la fuente de datos. Devolvería los datos a la base de datos federada y realizaría la clasificación localmente. Sin embargo, supongamos que la columna es un tipo de

datos de caracteres (CHAR o VARCHAR) y sólo contiene caracteres numéricos ('0','1',...,'9'). Puede indicar esto asignando un valor de 'Y' a la opción de columna de apodo NUMERIC_STRING. Esto proporciona al optimizador de consultas de DB2 la opción de realizar la clasificación en la fuente de datos. Si la clasificación se realiza remotamente, se evita la sobrecargar de llevar los datos al servidor federado y realizar la clasificación localmente.

Si se desean definir opciones de columna de apodo para apodos relacionales puede utilizarse la sentencia ALTER NICKNAME. Para apodos no relacionales, las opciones de columna de apodo pueden definirse utilizando las sentencias CREATE NICKNAME y ALTER NICKNAME.

Conceptos relacionados:

- "Correlaciones de tipos de datos" en la página 55

Tareas relacionadas:

- "Working with nicknames" en la publicación *Federated Systems Guide*

Información relacionada:

- "Opciones de la columna de apodo para sistemas federados" en la página 675

Correlaciones de tipos de datos

Los tipos de datos de la fuente de datos debe correlacionarse con los tipos de datos DB2 correspondientes, de forma que el servidor federado pueda recuperar los datos de las fuentes de datos. He aquí unos ejemplos de correlaciones de tipos de datos por omisión:

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

Para la mayor parte de las fuentes de datos, las correlaciones de tipos por omisión se encuentran en los reiniciadores. Las correlaciones de tipos por omisión para las fuentes de datos de DB2 se encuentran en el reiniciador DRDA. Las correlaciones de tipos por omisión para Informix® se encuentran en el reiniciador INFORMIX y así sucesivamente.

Para algunas fuentes de datos no relacionales, hay que especificar la información de tipos de datos en la sentencia CREATE NICKNAME. Deben especificarse los tipos de datos de DB2 para Linux, UNIX® y Windows® correspondientes para cada columna del objeto de la fuente de datos cuando se crea el apodo. Cada columna debe correlacionarse con una campo o una columna concreto del objeto de la fuente de datos.

Para las fuentes de datos relacionales, las correlaciones de tipos de datos por omisión pueden alterarse temporalmente. Por ejemplo, el tipo de datos INTEGER de Informix se correlaciona, por omisión, con el tipo de datos INTEGER de DB2. Es posible alterar temporalmente las correlaciones por omisión y correlacionar el tipo de datos INTEGER de Informix con el tipo de datos DECIMAL(10,0) de DB2.

Antes de crear un apodo, debe crear correlaciones de tipos nuevas o modificar las correlaciones de tipos por omisión. De lo contrario, los apodos creados antes de realizar los cambios en la correlación de tipos no reflejarán las nuevas correlaciones.

Conceptos relacionados:

- "Data type mappings in a federated system" en la publicación *Federated Systems Guide*

Correlaciones de funciones

Para que el servidor federado reconozca una función de la fuente de datos, la función debe estar correlacionada con una función complementaria existente en DB2® para Linux, UNIX® y Windows®. DB2 Information Integrator proporciona correlaciones por omisión entre funciones incorporadas de fuente de datos existentes y funciones incorporadas complementarias de DB2. Para la mayor parte de las fuentes de datos, las correlaciones de funciones por omisión se encuentran en los reiniciadores. Las correlaciones de funciones por omisión con funciones DB2 para z/OS™ y OS/390® se encuentran en el reiniciador DRDA®. Las correlaciones de funciones por omisión con funciones de Sybase se encuentran en el reiniciador CTLIB y así sucesivamente.

Para fuentes de datos relacionales, es posible crear una correlación de funciones cuando se desea utilizar una función de la fuente de datos que el servidor federado no reconozca. La correlación que se crea es entre la función de la fuente de datos y una función complementaria de DB2 situada en la base de datos federada. Las correlaciones de funciones suelen utilizarse cuando en la fuente de datos existe una nueva función incorporada o una nueva función definida por el usuario. Las correlaciones de funciones también se utilizan cuando no existe una función complementaria de DB2. En este caso, también se debe crear una plantilla de función.

Conceptos relacionados:

- "Function mappings in a federated system" en la publicación *Federated Systems Guide*
- "Especificaciones de índice" en la página 56

Especificaciones de índice

Cuando crea un apodo para una tabla de la fuente de datos, se añade información sobre los índices que tenga la tabla de la fuente de datos al catálogo global. El optimizador de consultas utiliza esta información para acelerar el proceso de las peticiones distribuidas. La información de catálogo sobre un índice de la fuente de datos es un conjunto de metadatos y se denomina una *especificación de índice*. Un servidor federado no crea una especificación de índice cuando se crea un apodo para:

- Una tabla que no tiene índices
- Una vista, que no suele tener almacenada ninguna información sobre los índices en el catálogo remoto
- Un objeto de la fuente de datos que no tiene un catálogo remoto a partir del cual el servidor federado pueda obtener la información sobre los índices

Supongamos que una tabla adquiere un índice nuevo además de los que tenía cuando se creó el apodo. Como la información sobre índices se proporciona al catálogo global en el momento en que se crea el apodo, el servidor federado desconoce la existencia del índice nuevo. De forma similar, cuando se crea un apodo para una vista, el servidor federado no es consciente de la tabla subyacente (y los índices de la misma) a partir de la cual se ha generado la vista. En estas circunstancias, es posible suministrar la información sobre índices necesaria al

catálogo global. Puede crear una especificación de índice para las tablas que no tienen índices. La especificación de índice indica al optimizador de la consulta en qué columna o columnas de la tabla debe buscar para buscar los datos con rapidez.

Conceptos relacionados:

- "Index specifications in a federated system" en la publicación *Federated Systems Guide*

Orden de clasificación

El orden en el que los datos de caracteres se clasifican en una base de datos depende de la estructura de los datos y del orden de clasificación definido para la base de datos.

Supongamos que todos los datos de una base de datos son caracteres en mayúsculas y no contienen ningún carácter numérico ni especial. Una clasificación de los datos debería generar el mismo resultado con independencia de si los datos se clasifican en la fuente de datos o en la base de datos federada. El orden de clasificación que cada base de datos utilice no debería tener ningún efecto sobre el resultado de la clasificación. Del mismo modo, si todos los datos de la base de datos son caracteres en minúsculas o son caracteres numéricos, una clasificación de los datos debería generar el mismo resultado con independencia del lugar en que se realice la clasificación.

Si los datos están formados por una de las estructuras siguientes:

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

La clasificación de estos datos puede generar distintos resultados si la base de datos federada y la fuente de datos utilizan un orden de clasificación distinto.

En general, un *orden de clasificación* es un orden definido para los datos de caracteres que determina si un carácter determinado se clasifica por encima, por debajo o al mismo nivel que otro carácter.

Forma en que un orden de clasificación determina el orden de clasificación de los caracteres

Un orden de clasificación determina el orden de clasificación de los caracteres de un conjunto de caracteres codificados. Un *conjunto de caracteres* es el conjunto de caracteres que se utilizan en un sistema informático o en un lenguaje de programación. En un conjunto de caracteres *codificados* a cada carácter se le asigna un número distinto comprendido entre 0 y 255 (o su equivalente hexadecimal). Los números se denominan *elementos de código*; las asignaciones entre números y caracteres dentro de un conjunto se denominan colectivamente una *página de códigos*.

Además de asignarse a un carácter, un elemento de código puede correlacionarse con la posición de un carácter en un orden de clasificación. En términos técnicos, un orden de clasificación es, por lo tanto, la correlación colectiva de los elementos de código de un conjunto de caracteres con las posiciones del orden de clasificación de los caracteres del conjunto. La posición de un carácter se representa

mediante un número; este número se denomina el *peso* del carácter. En el orden de clasificación más simple, denominado un *orden de identidad*, los pesos son idénticos a los elementos de código.

Supongamos que la base de datos ALPHA utiliza el orden de clasificación por omisión de la página de códigos EBCDIC y que la base de datos BETA utiliza el orden de clasificación por omisión de la página de códigos ASCII. El orden de clasificación de las series de caracteres de estas dos bases de datos sería distinto, tal como se muestra en el ejemplo siguiente:

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

Clasif. basada en EBCDIC    Clasif. basada en ASCII

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

De forma similar, las comparaciones de caracteres en una base de datos dependen del orden de clasificación definido para dicha base de datos. En este ejemplo, la base de datos ALPHA utiliza el orden de clasificación por omisión de la página de códigos EBCDIC. La base de datos BETA utiliza el orden de clasificación por omisión de la página de códigos ASCII. Las comparaciones de los caracteres de estas dos bases de datos generarían resultados distintos, tal como se muestra en el ejemplo siguiente:

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

Resultado basado en EBCDIC   Resultado basado en ASCII
COL2                         COL2
----                         ----
TW4                          TW4
X82                          X82
39G
```

Definición del orden de clasificación local para optimizar las consultas

Los administradores pueden crear bases de datos federadas con un orden de clasificación concreto que coincida con un orden de clasificación de fuente de datos. Entonces, para cada definición del servidor de la fuente de datos, la opción del servidor COLLATING_SEQUENCE se establece en 'Y'. Este valor indica a la base de datos federada que el orden de clasificación de la base de datos federada y de la fuente de datos debe coincidir.

El orden de clasificación de la base de datos federada se establece como parte de la API CREATE DATABASE. Mediante esta API, es posible especificar uno de los órdenes siguientes:

- Un orden de identidad
- Un orden del *sistema* (el orden utilizado por el sistema operativo que proporciona soporte a la base de datos)
- Un orden *personalizado* (un orden predefinido que DB2 UDB proporciona o que el propio usuario define)

Supongamos que la fuente de datos es DB2 para z/OS y OS/390. Un orden de clasificación basado en una página de códigos EBCDIC implementa las

clasificaciones definidas en una cláusula ORDER BY. Para recuperar datos de DB2 para z/OS y OS/390 clasificados de acuerdo a cláusulas ORDER BY, la base de datos federada debe configurarse de forma que utilice el orden de clasificación predefinido basado en la página de códigos EBCDIC adecuada.

Conceptos relacionados:

- "Server characteristics affecting pushdown opportunities" en la publicación *Federated Systems Guide*
- "Secuencias de clasificación en un sistema federado" en la publicación *IBM DB2 Information Integrator Guía de configuración de fuentes de datos*

Tareas relacionadas:

- "Creación de una base de datos federada" en la publicación *IBM DB2 Information Integrator Guía de configuración de fuentes de datos*

Información relacionada:

- "National language versions" en la publicación *Administration Guide: Planning*
- "Consideraciones sobre idiomas nacionales de bases de datos federadas" en la publicación *IBM DB2 Information Integrator Guía de configuración de fuentes de datos*

Capítulo 2. Elementos del lenguaje

Este capítulo describe los elementos del lenguaje que son comunes a muchas sentencias de SQL:

Caracteres

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

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

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

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

Carácter	Descripción	Carácter	Descripción
	espacio o blanco	–	signo menos
"	apóstrofo, comillas simples o comillas dobles	.	punto
%	porcentaje	/	barra inclinada
&	signo &	:	dos puntos
'	apóstrofo o comillas simples	;	punto y coma
(	abrir paréntesis	<	menor que
)	cerrar paréntesis	=	igual
*	asterisco	>	mayor que
+	signo más	?	signo de interrogación
,	coma	_	subrayado

Caracteres

Carácter	Descripción	Carácter	Descripción
|	barra vertical[1]	^	signo de intercalación
!	signo de admiración	[	abrir corchete
{	abrir llave	]	cerrar corchete
}	cerrar llave		

[1] La utilización del carácter de barra vertical (|) podría impedir la portabilidad de los códigos entre productos relacionales de IBM. Debe utilizarse el operador CONCAT en lugar del operador ||.

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

Símbolos

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

Los símbolos se clasifican en ordinarios y delimitadores:

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

  Ejemplos

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

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

  Ejemplos

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

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

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

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

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

después de la conversión, es equivalente a:

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

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

Información relacionada:

- "Cómo se invocan las sentencias de SQL" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia PREPARE" en la publicación *Consulta de SQL, Volumen 2*

Identificadores

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

- identificadores de SQL

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

  – Un *identificador ordinario* es una letra seguida de cero o más caracteres, cada uno de los cuales puede ser una letra mayúscula, un dígito o el carácter de subrayado. Un identificador ordinario no debe ser idéntico a una palabra reservada.

    Ejemplos

    ```
    WKLYSAL     WKLY_SAL
    ```

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

    Ejemplos

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

  Las conversión de caracteres de los identificadores creados en una página de códigos de doble byte pero utilizados por una aplicación o una base de datos en una página de códigos de múltiples bytes pueden necesitar una consideración especial.

- Identificadores del lenguaje principal

  Un *identificador del lenguaje principal* es un nombre declarado en el programa de lenguaje principal. Las reglas para formar un identificador de lenguaje principal son las reglas del lenguaje principal. Un identificador de lenguaje principal no debe tener una longitud mayor que 255 caracteres y no debe comenzar con los caracteres 'SQL' ni 'DB2' (ni en mayúsculas ni en minúsculas).

Convenios de denominación y calificaciones de nombre de objeto implícitas

Las reglas para formar el nombre de un objeto dependen del tipo de objeto. Los nombres de los objetos de una base de datos pueden constar de un solo identificador o pueden ser objetos calificados mediante esquema que consten de dos identificadores. Los nombres de los objetos calificados mediante esquema pueden especificarse sin el nombre de esquema; en este caso, el nombre de esquema es implícito.

En las sentencias de SQL dinámico, un nombre de objeto calificado con un esquema utiliza implícitamente el valor de registro especial CURRENT SCHEMA como calificador para las referencias al nombre de objeto no calificadas. Por omisión, se establece en el ID de autorización actual. Si la sentencia de SQL dinámico está contenida en un paquete que muestra un comportamiento de vinculación, definición o invocación, el registro especial CURRENT SCHEMA no se utiliza en la calificación. En un paquete con comportamiento de vinculación, se utiliza el calificador por omisión del paquete como el valor para la calificación implícita de las referencias al objeto no calificadas. En un paquete con comportamiento de definición, se utiliza el ID de autorización de la persona que define la rutina como el valor para la calificación implícita de las referencias al objeto no calificadas en la rutina. En un paquete con comportamiento de invocación, se utiliza el ID en vigor al invocar la rutina como el valor para la

calificación implícita de las referencias al objeto no calificadas en las sentencias de SQL dinámico de la rutina. Para obtener más información, consulte el apartado "Características de SQL dinámico durante la ejecución" en la página 71.

En las sentencias de SQL estático, la opción de precompilación/vinculación QUALIFIER especifica implícitamente el calificador para los nombres de objetos de base de datos no calificados. Por omisión, este valor se establece en el ID de autorización del paquete.

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

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

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

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

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

nombre-autorización Identificador que designa un usuario o un grupo:

- Los caracteres válidos van de la "A" a la "Z", de la "a" a la "z", del 0 al 9, #, @, $ y _.
- El nombre no debe empezar por los caracteres 'SYS', 'IBM' ni 'SQL'.
- El nombre no debe ser: ADMINS, GUESTS, LOCAL, PUBLIC ni USERS.
- Un ID de autorización delimitado no debe contener letras en minúsculas.

nombre-agrupalmacinter Identificador que designa una agrupación de almacenamientos intermedios.

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

nombre-condición Identificador que designa una condición en un procedimiento de SQL.

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

nombre-correlación Identificador que designa una tabla resultante.

nombre-cursor Identificador que designa un cursor de SQL. Para

	compatibilidad entre sistemas principales, se puede utilizar un carácter de guión en el nombre.
nombre-fuente-datos	Identificador que designa una fuente de datos. Este identificador es la primera de las tres partes de un nombre de objeto remoto.
nombre-descriptor	Dos puntos seguidos de un identificador del lenguaje principal que designa un área de descriptores de SQL (SQLDA). Para ver la descripción de un identificador de lenguaje principal,consulte el apartado “Referencias a variables del lenguaje principal” en la página 79. Observe que un nombre de descriptor nunca incluye una variable indicadora.
nombre-tipo-diferenciado	Nombre calificado o no calificado que designa un tipo diferenciado. El gestor de bases de datos califica implícitamente un nombre de tipo diferenciado no calificado en una sentencia de SQL según el contexto.
nombre-supervisor-sucesos	Identificador que designa un supervisor de sucesos.
nombre-correlación-funciones	Identificador que designa una correlación de funciones.
nombre-función	Nombre calificado o no calificado que designa una función. El gestor de bases de datos califica implícitamente un nombre de función no calificado en una sentencia de SQL según el contexto.
nombre-grupo	Identificador no calificado que designa un grupo de transformación definido para un tipo estructurado.
variable-lengprinc	Secuencia de símbolos que designa una variable del lenguaje principal. Una variable del lenguaje principal incluye, como mínimo, un identificador de lenguaje principal, como se explica en el apartado “Referencias a variables del lenguaje principal” en la página 79.
nombre-índice	Nombre calificado mediante esquema que designa un índice o una especificación de índice.
etiqueta	Identificador que designa una etiqueta en un procedimiento de SQL.
nombre-método	Identificador que designa un método. El contexto de esquema de un método está determinado por el esquema del tipo indicado (o de un supertipo del tipo indicado) del método.
apodo	Nombre calificado mediante esquema que designa una referencia de servidor federado a una tabla o vista.
nombre-grupo-particiones-bd	Identificador que designa un grupo de particiones de base de datos.
nombre-paquete	Nombre calificado mediante esquema que designa

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

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

nombre-procedimiento Nombre calificado o no calificado que designa un procedimiento. El gestor de bases de datos califica implícitamente un nombre de procedimiento no calificado en una sentencia de SQL según el contexto.

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

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

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

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

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

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

nombre-puntosalvar Identificador que designa un punto de salvar.

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

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

	Para evitar complicaciones, es recomendable no utilizar el nombre SESSION como esquema, excepto para el esquema de tablas temporales globales declaradas (las cuales *deben* utilizar el nombre de esquema SESSION).
nombre-servidor	Identificador que designa un servidor de aplicaciones. En un sistema federado, el nombre de servidor también designa el nombre local de una fuente de datos.
nombre-específico	Nombre, calificado o no calificado, que designa un nombre específico. El gestor de bases de datos califica implícitamente un nombre específico no calificado en una sentencia de SQL según el contexto.
nombre-variable-SQL	El nombre de una variable local en una sentencia de procedimiento SQL. Los nombres de variables SQL se pueden utilizar en otras sentencias de SQL donde esté permitido un nombre de variable del lenguaje principal. El nombre puede estar calificado por la etiqueta de la sentencia compuesta donde se declaró la variable de SQL.
nombre-sentencia	Identificador que designa una sentencia de SQL preparada.
nombre-supertipo	Nombre calificado o no calificado que designa el supertipo de un tipo. El gestor de bases de datos califica implícitamente un nombre de supertipo no calificado en una sentencia de SQL según el contexto.
nombre-tabla	Nombre calificado mediante esquema que designa una tabla.
nombre-espacio-tablas	Identificador que designa un espacio de tablas.
nombre-activador	Nombre calificado mediante esquema que designa un activador.
nombre-correlación-tipos	Identificador que designa una correlación de tipos de datos.
nombre-tipo	Nombre calificado o no calificado que designa un tipo. El gestor de bases de datos califica implícitamente un nombre de tipo no calificado en una sentencia de SQL según el contexto.
nombre-tabla-tipo	Nombre calificado mediante esquema que designa una tabla con tipo.
nombre-vista-tipo	Nombre calificado mediante esquema que designa una vista con tipo.
nombre-vista	Nombre calificado mediante esquema que designa una vista.
nombre-reiniciador	Identificador que designa un reiniciador.

Seudónimos

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

Un seudónimo se puede utilizar siempre que se pueda utilizar el nombre de una tabla o una vista. Se puede crear un seudónimo aunque no exista el objeto (aunque debe existir en el momento de compilar una sentencia que hace referencia al mismo). Puede hacer referencia a otro seudónimo si no se realizan referencias circulares ni repetitivas a lo largo de la cadena de seudónimos. Un seudónimo sólo puede hacer referencia a una tabla, una vista o un seudónimo de la misma base de datos. Un seudónimo no se puede utilizar cuando se espera un nombre de tabla o de vista nuevo como, por ejemplo, en las sentencias CREATE TABLE o CREATE VIEW; por ejemplo, si se ha creado el seudónimo PERSONAL, las sentencias posteriores como, por ejemplo, CREATE TABLE PERSONAL... devolverán un error.

La opción de para hacer referencia a una tabla o una vista mediante un seudónimo no se muestra explícitamente en los diagramas de sintaxis, ni se menciona en las descripciones de las sentencias de SQL.

Un seudónimo no calificado nuevo no puede tener el mismo nombre completamente calificado que una tabla, una vista o un seudónimo existente.

El efecto de utilizar un seudónimo en una sentencia de SQL es similar al de la sustitución de texto. El seudónimo, que debe definirse antes de compilar la sentencia de SQL, se sustituye en el momento de compilar la sentencia por el nombre base calificado de la tabla o vista. Por ejemplo, si PBIRD.SALES es un seudónimo de DSPN014.DIST4_SALES_148, entonces en el momento de la compilación:

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

se convierte en realidad en

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

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

En la tolerancia de sintaxis de las aplicaciones que se ejecutan bajo otros sistemas de gestión de bases de datos relacionales, se puede utilizar SYNONYM en vez de ALIAS en las sentencias CREATE ALIAS y DROP ALIAS.

ID de autorización y nombres de autorización

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

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

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

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

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

Para obtener más información, consulte el apartado “Características de SQL dinámico durante la ejecución” en la página 71.

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

Ejemplos:

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

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

  SMITH es el ID de autorización de la sentencia. Por lo tanto, en una sentencia de SQL dinámico, el valor por omisión del registro especial CURRENT SCHEMA será SMITH y, en SQL estático, el valor por omisión de la opción de precompilación/vinculación QUALIFIER será SMITH. La autorización para ejecutar la sentencia se compara con SMITH y SMITH es el calificador implícito de *nombre-tabla* de acuerdo con las reglas de calificación descritas en el apartado “Convenios de denominación y calificaciones de nombre de objeto implícitas” en la página 64.

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

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

  ```
  DROP TABLE TDEPT
  ```

  Elimina la tabla SMITH.TDEPT.

  ```
  DROP TABLE SMITH.TDEPT
  ```

  Elimina la tabla SMITH.TDEPT.

  ```
  DROP TABLE KEENE.TDEPT
  ```

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

  ```
  CREATE SCHEMA PAYROLL AUTHORIZATION KEENE
  ```

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

Características de SQL dinámico durante la ejecución

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

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

Tabla 4. Forma en que DYNAMICRULES y el entorno de ejecución determinan el comportamiento de las sentencias de SQL dinámico

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

Comportamiento de ejecución

DB2 utiliza el ID de autorización del usuario (el ID que inicialmente se conectó a DB2) que ejecuta el paquete como el valor que se utilizará para la comprobación de autorización de las sentencias de SQL dinámico y para el valor inicial utilizado para la calificación implícita de las referencias a objetos no calificadas de las sentencias de SQL dinámico.

Comportamiento de vinculación

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

Comportamiento de definición

El comportamiento de definición sólo se aplica si la sentencia de SQL dinámico está en un paquete que se ejecuta en un contexto de rutina y el paquete se ha vinculado con DYNAMICRULES DEFINEBIND o DYNAMICRULES DEFINERUN. DB2 utiliza el ID de autorización de la persona que define la rutina (no de la que vincula el paquete de la rutina) como el valor que se utilizará para la comprobación de autorización de las sentencias de SQL dinámico y para la calificación implícita de las referencias a objetos no calificadas de sentencias de SQL dinámico de dicha rutina.

Comportamiento de invocación

El comportamiento de invocación sólo se aplica si la sentencia de SQL dinámico está en un paquete que se ejecuta en un contexto de rutina y el paquete se ha vinculado con DYNAMICRULES INVOKEBIND o DYNAMICRULES INVOKERUN. DB2 utiliza el ID de autorización de la sentencia en vigor cuando se invoca la rutina como el valor que se utilizará para la comprobación de autorización de SQL dinámico y para la calificación implícita de las referencias a objetos no calificadas de sentencias de SQL dinámico de dicha rutina. La tabla siguiente muestra un resumen.

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

Sentencias restringidas cuando no se aplica el comportamiento de ejecución

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

Consideraciones respecto a la opción DYNAMICRULES

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

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

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

ID de autorización y preparación de sentencias

Si se especifica la opción VALIDATE BIND durante la ejecución, los privilegios necesarios para manejar tablas y vistas también deben existir durante la vinculación. Si estos privilegios o los objetos referenciados no existen y está en vigor la opción SQLERROR NOPACKAGE, la operación de vinculación no será satisfactoria. Si se especifica la opción SQLERROR CONTINUE, la operación de vinculación será satisfactoria y se marcarán las sentencias erróneas. Si se intenta ejecutar una de estas sentencias, se producirá un error.

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

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

Nombres de columna

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

- Declarar el nombre de una columna como, por ejemplo, en una sentencia CREATE TABLE.
- Identificar una columna como, por ejemplo, en una sentencia CREATE INDEX.

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

Nombre de columna calificados

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

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

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

Cuando un calificador es opcional, puede cumplir dos finalidades. Estos casos se describen en el apartado “Calificadores de nombres de columna para evitar ambigüedades” en la página 76 y “Calificadores de nombres de columna en referencias correlacionadas” en la página 78.

Nombres de correlación

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

```
FROM X.MYTABLE Z
```

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

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

Como calificador, un nombre de correlación puede utilizarse para evitar ambigüedades o para establecer una referencia correlacionada. También puede

utilizarse simplemente como nombre abreviado de una tabla, vista, apodo o seudónimo. En el caso de una expresión de tabla anidada o una función de tabla, es necesario un nombre de correlación para identificar la tabla resultante. En el ejemplo, Z podría haberse utilizado simplemente para evitar tener que entrar X.MYTABLE más de una vez.

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

Ejemplo

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

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

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

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

```
FROM EMPLOYEE E, DEPARTMENT
```

Un nombre de tabla, vista, apodo o seudónimo que está expuesto en una cláusula FROM puede ser igual a otro nombre de tabla, vista o apodo expuesto en esa cláusula FROM o cualquier nombre de correlación de la cláusula FROM. Esta situación puede dar como resultado una serie de referencias ambiguas de nombres de columna que acaban devolviendo un código de error (SQLSTATE 42702).

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

1. Dada una cláusula FROM:

   ```
   FROM EMPLOYEE E1, EMPLOYEE
   ```

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

   ```
   FROM EMPLOYEE, EMPLOYEE E2
   ```

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

   ```
   FROM EMPLOYEE, EMPLOYEE
   ```

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

4. Dada la sentencia siguiente:

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

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

5. Dada una cláusula FROM:

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

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

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

Dada una cláusula FROM:

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

una referencia calificada como, por ejemplo, D.NUM indica la primera columna de la tabla DEPARTMENT que se ha definido en la tabla como DEPTNO. Una referencia a D.DEPTNO utilizando esta cláusula FROM es incorrecta ya que el nombre de columna DEPTNO es un nombre de columna no expuesto.

Calificadores de nombres de columna para evitar ambigüedades

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

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

Designadores de tablas: Un calificador que designa una tabla de objeto específica se conoce como *designador de tabla*. La cláusula que identifica las tablas de objetos

también establece los designadores de tabla para ellas. Por ejemplo, las tablas de objetos de una expresión en una cláusula SELECT se nombran en la cláusula FROM que la sigue:

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

Los designadores en la cláusula FROM se establecen como sigue:

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

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

Evitar referencias no definidas o ambiguas: Cuando un nombre de columna hace referencia a valores de una columna, debe existir una sola tabla de objetos que incluya una columna con ese nombre. Las situaciones siguientes se consideran errores:

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

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

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

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

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

es una sentencia válida.

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

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

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

Calificadores de nombres de columna en referencias correlacionadas

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

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

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

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

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

Cuando el nombre de columna de una subconsulta no se califica, se busca en las tablas a las que se hace referencia en cada nivel de la jerarquía, empezando en la misma subconsulta en la que aparece el nombre de columna y siguiendo hacia

niveles superiores de la jerarquía hasta que se encuentre un nombre de columna que coincida. Si la columna se encuentra en una tabla en un nivel superior al nivel que contiene el nombre de columna, es que éste es una referencia correlacionada para el nivel donde se ha encontrado la tabla que contiene la columna. Si se encuentra el nombre de columna en más de una tabla en un nivel en concreto, la referencia es ambigua y se considera un error.

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

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

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

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

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

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

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

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

Referencias a variables del lenguaje principal

Una *variable del lenguaje principal* es:

- Una variable de un lenguaje principal como por ejemplo, una variable C, una variable C++, un elemento de datos COBOL, una variable FORTRAN o una variable Java

o:

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

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

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

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

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

Variables del lenguaje principal en SQL dinámico

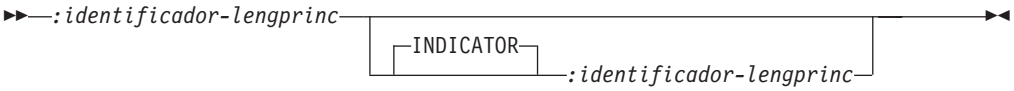
En sentencias de SQL dinámico, se utilizan los marcadores de parámetros en lugar de las variables del lenguaje principal. Un marcador de parámetros es un signo de interrogación (?) que representa una posición en una sentencia de SQL dinámico en la que la aplicación proporcionará un valor; es decir, donde se encontrará una variable del lenguaje principal si la serie de la sentencia es una sentencia de SQL estático. El siguiente ejemplo muestra una sentencia de SQL estático que emplea variables del lenguaje principal:

```
INSERT INTO DEPARTMENT
   VALUES (:hv_deptno, :hv_deptname, :hv_mgrno, :hv_admrdept)
```

Este ejemplo muestra una sentencia de SQL dinámico que utiliza marcadores de parámetros:

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

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



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

El primer identificador-lengprinc designa la *variable principal*. Según la operación, proporciona un valor al gestor de bases de datos o bien el gestor de bases de datos le proporciona un valor. Una variable del lenguaje principal de entrada

proporciona un valor en la página de códigos de la aplicación en tiempo de ejecución. A la variable del lenguaje principal de salida se le proporciona un valor que, si es necesario, se convierte a la página de códigos de la aplicación en tiempo de ejecución cuando los datos se copian en la variable de la aplicación de salida. Una variable del lenguaje principal determinada puede servir tanto de variable de entrada como de salida en el mismo programa.

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

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

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

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

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

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

Ejemplo: Utilizando la tabla PROJECT, asigne a la variable del lenguaje principal PNAME (VARCHAR(26)) el nombre de proyecto (PROJNAME), a la variable del lenguaje principal STAFF (dec(5,2)) el nivel principal de personal (PRSTAFF) y a la variable del lenguaje principal MAJPROJ (char(6)) el proyecto principal (MAJPROJ) para el proyecto (PROJNO) 'IF1000'. Las columnas PRSTAFF y MAJPROJ pueden

contener valores nulos, por lo tanto proporcione las variables indicadoras STAFF_IND (smallint) y MAJPROJ_IND (smallint).

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

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

Referencias a las variables del lenguaje principal de BLOB, CLOB y DBCLOB

Las variables regulares BLOB, CLOB y DBCLOB, las variables localizadoras LOB (consulte "Referencias a variables localizadoras"), y las variables de referencia a archivos LOB (consulte "Referencias a las variables de referencia de archivos BLOB, CLOB y DBCLOB" en la página 83) se pueden definir en todos los lenguajes principales. Donde se pueden utilizar valores LOB, el término *variable-lengprinc* en un diagrama de sintaxis puede hacer referencia a una variable del lenguaje principal normal, a una variable localizadora o a una variable de referencia a archivos. Puesto que no son tipos de datos nativos, se utilizan las extensiones SQL y los precompiladores generan las construcciones de lenguaje principal necesarias para poder representar a cada variable. En cuanto a REXX, las variables LOB se correlacionan con series.

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

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

Referencias a variables localizadoras

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

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

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

Cuando la variable indicadora asociada a un localizador es nula, el valor del LOB al que se hace referencia también es nulo.

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

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

Referencias a las variables de referencia de archivos BLOB, CLOB y DBCLOB

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

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

Una variable de referencia a archivos tiene las siguientes propiedades:

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

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

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

Nombre del archivo
Debe especificarlo el programa de aplicación en tiempo de ejecución. Puede ser:

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

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

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

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

de la variable de referencia a archivos. Se debe especificar alguna de estas opciones para cada variable de referencia a archivos:

- Entrada (de cliente a servidor)

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

- Salida (de servidor a cliente)

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

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

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

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

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

Ejemplo de una variable de referencia a archivos de salida (en C): Supongamos una sección de declaración codificada como:

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

Una vez procesada:

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

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

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

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

Ejemplo de una variable de referencia a archivos de entrada (en C): Tomando la misma sección de declaración que antes, se puede utilizar el siguiente código para insertar datos de un archivo normal al que :hv_text_file hace referencia en una columna CLOB.

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

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

Referencias a variables del lenguaje principal de tipo estructurado

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

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

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

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

Referencias a variables del lenguaje principal de tipo estructurado

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

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

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

Información relacionada:

- "Sentencia CREATE ALIAS" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia PREPARE" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia SET SCHEMA" en la publicación *Consulta de SQL, Volumen 2*
- Apéndice A, "Límites de SQL", en la página 525
- Apéndice C, "SQLDA (área de descriptores de SQL)", en la página 539
- Apéndice G, "Nombres de esquema reservados y palabras reservadas", en la página 741
- Apéndice N, "Consideraciones sobre el código UNIX ampliado (EUC) en japonés y chino tradicional", en la página 801
- "Consultas de SQL" en la página 469
- "Objeto grande (LOB)" en la página 94

Tipos de datos

Tipos de datos

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

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

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

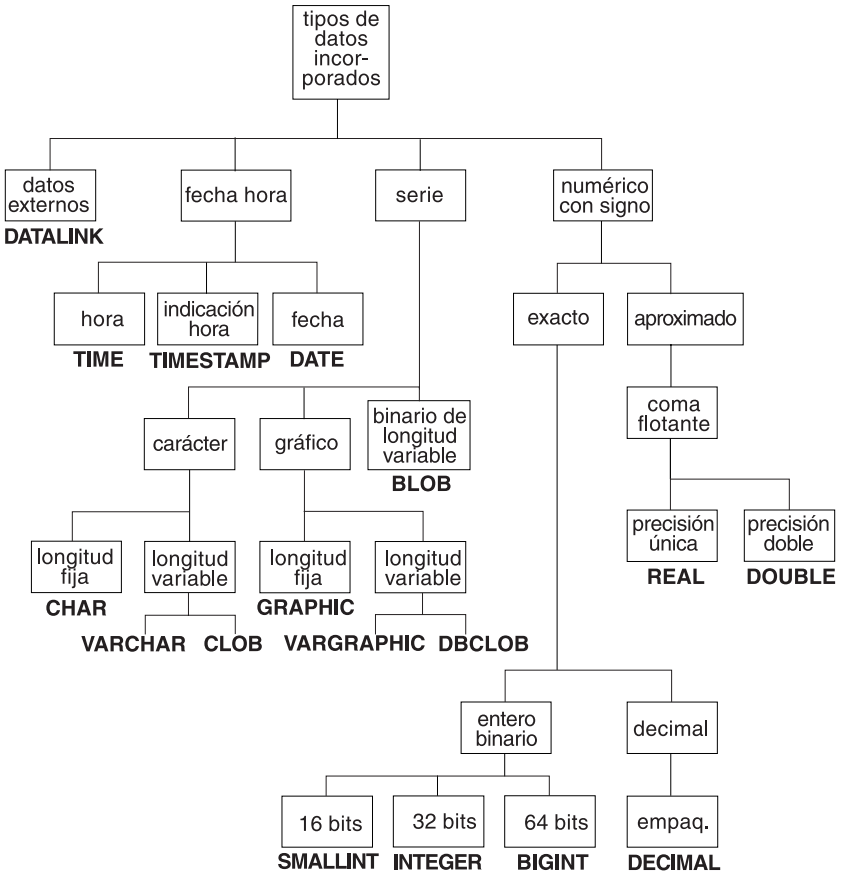


Figura 10. Tipos de datos internos de DB2 soportados

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

Información relacionada:

- "Tipos definidos por el usuario" en la página 102

Números

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

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

Entero pequeño (SMALLINT)

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

Entero grande (INTEGER)

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

Entero superior (BIGINT)

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

Coma flotante de precisión simple (REAL)

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

Coma flotante de doble precisión (DOUBLE o FLOAT)

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

Decimal (DECIMAL o NUMERIC)

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

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

Información relacionada:

- "Sentencia CREATE TABLE" en la publicación *Consulta de SQL, Volumen 2*
- Apéndice C, "SQLDA (área de descriptores de SQL)", en la página 539

Series de caracteres

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

Serie de caracteres de longitud fija (CHAR)

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

Series de caracteres de longitud variable

Existen tres tipos de series de caracteres de longitud variable:

- Un valor VARCHAR puede tener una longitud máxima de 32 672 bytes.
- Un valor LONG VARCHAR puede tener una longitud máxima de 32 700 bytes.
- Un valor CLOB (objeto grande de caracteres) puede tener una longitud máxima de 2 gigabytes (2 147 483 647 bytes). Un CLOB se utiliza para almacenar datos basados en caracteres SBCS o mixtos (SBCS y MBCS) (como, por ejemplo, documentos escritos con un solo juego de caracteres) y, por lo tanto, tiene una página de códigos SBCS o mixta asociada).

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

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

Las funciones del esquema SYSFUN que toman como argumento VARCHAR no aceptarán las VARCHAR que tengan más de 4 000 bytes de longitud como argumento. Sin embargo, muchas de estas funciones también pueden tener una signatura alternativa que acepte un CLOB(1M). Para estas funciones, el usuario puede convertir explícitamente las series VARCHAR mayores que 4 000 en datos CLOB y luego reconvertir el resultado en datos VARCHAR de la longitud deseada.

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

Cada serie de caracteres se define con más detalle como:

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

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

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

Series gráficas

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

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

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

Series gráficas de longitud fija (GRAPHIC)

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

Series gráficas de longitud variable

Existen tres tipos de series gráficas de longitud variable:

- Un valor VARGRAPHIC puede tener una longitud máxima de 16 336 caracteres de doble byte.
- Un valor LONG VARGRAPHIC puede tener una longitud máxima de 16 350 caracteres de doble byte.
- Un valor DBCLOB (objeto grande de caracteres de doble byte) puede tener una longitud máxima de 1 073 741 823 caracteres de doble byte. Un DBCLOB se utiliza para almacenar datos DBCS grandes basados en caracteres (por ejemplo, documentos escritos con un solo juego de caracteres) y, por lo tanto, tiene asociada una página de códigos DBCS).

Se aplican restricciones especiales a una expresión que dé como resultado una serie gráfica de longitud variable cuya longitud máxima sea mayor que 127 bytes. Estas restricciones son las mismas que las especificadas en el apartado “Series de caracteres de longitud variable” en la página 90.

Series binarias

Una *serie binaria* es una secuencia de bytes. A diferencia de las series de caracteres, que suelen contener datos de texto, las series binarios se utilizan para contener datos no tradicionales como, por ejemplo, imágenes, voz o soportes mixtos. Las series de caracteres del subtipo FOR BIT DATA puede utilizarse para fines similares, pero los dos tipos de datos no son compatibles. La función escalar BLOB puede utilizarse para convertir una serie de caracteres FOR BIT DATA en una serie binaria. Las series binarias no están asociadas a ninguna página de códigos. Tienen las mismas restricciones que las series de caracteres (consulte los detalles en el apartado "Series de caracteres de longitud variable" en la página 90).

Objeto grande binario (BLOB)

Un *objeto grande binario* es una serie binario de longitud variable que puede tener una longitud máxima de 2 gigabytes (2 147 483 647 bytes). Los valores BLOB pueden contener datos estructurados para que los utilicen las funciones definidas por el usuario y los tipos definidos por el usuario. Igual que las series de caracteres FOR BIT DATA, las series BLOB no están asociadas a ninguna página de códigos.

Objeto grande (LOB)

El término *objeto grande* y el acrónimo genérico LOB hace referencia al tipo de datos BLOB, CLOB o DBCLOB. Los valores LOB están sujetos a las restricciones que se aplican a los valores LONG VARCHAR, que se describen en el apartado "Series de caracteres de longitud variable" en la página 90. Estas restricciones se aplican incluso si el atributo de longitud de la serie LOB es de 254 bytes o menor.

Los valores LOB pueden ser muy grandes y la transferencia de dichos valores desde servidor de bases de datos a las variables del lenguaje principal del programa de aplicación cliente puede tardar mucho tiempo. Como normalmente los programas de aplicación procesan los valores LOB de fragmento en fragmento en lugar de como un todo, las aplicaciones pueden hacer referencia a un valor LOB utilizando un localizador de objeto grande.

Un *localizador de objeto grande* o localizador de LOB es una variable del lenguaje principal cuyo valor representa un solo valor LOB del servidor de bases de datos.

Un programa de aplicación puede seleccionar un valor LOB en un localizador de LOB. Entonces, utilizando el localizador de LOB, el programa de aplicación puede solicitar operaciones de base de datos basadas en el valor LOB (por ejemplo, aplicar las funciones escalares SUBSTR, CONCAT, VALUE o LENGTH, realizar una asignación, efectuar búsquedas en el LOB con LIKE o POSSTR o aplicar funciones definidas por el usuario sobre el LOB) proporcionando el valor del localizador como entrada. La salida resultante (los datos asignados a una variable del lenguaje principal cliente), sería normalmente un subconjunto pequeño del valor LOB de entrada.

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

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

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

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

Un localizador de LOB es sólo un mecanismo utilizado para hacer referencia a un valor LOB durante una transacción; no persiste más allá de la transacción en la que se ha creado. No es un tipo de base de datos; nunca se almacena en la base de datos y, como resultado, no puede participar en vistas ni en restricciones de comprobación. Sin embargo, como un localizador de LOB es una representación cliente de un tipo LOB, hay SQLTYPE para localizadores de LOB para que puedan describirse dentro de una estructura SQLDA que se utiliza por sentencias FETCH, OPEN y EXECUTE.

Valores de fecha y hora

Entre los tipos de datos de fecha y hora se incluyen DATE, TIME y TIMESTAMP. Aunque los valores de fecha y hora se pueden utilizar en algunas operaciones aritméticas y de series y son compatibles con algunas series, no son ni series ni números.

Fecha

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

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

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

Hora

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

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

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

Indicación de fecha y hora

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

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

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

Representación mediante series de los valores de fecha y hora

Los valores cuyos tipos de datos son DATE, TIME o TIMESTAMP se representan en un formato interno que es transparente para el usuario. Sin embargo, los valores de fecha, hora e indicación de fecha y hora también pueden representarse mediante series. Esto resulta útil porque no existen constantes ni variables cuyo tipo de datos sean DATE, TIME o TIMESTAMP. Antes de poder recuperar un valor

de fecha y hora, éste debe asignarse a una variable de serie. La función GRAPHIC (sólo para bases de datos Unicode) puede utilizarse para cambiar el valor de fecha y hora a una representación de serie. Normalmente, la representación de serie es el formato por omisión de los valores de fecha y hora asociados con el código territorial de la aplicación, a menos que se alteren temporalmente por la especificación de la opción DATETIME al precompilar el programa o vincularlo con la base de datos.

Con independencia de su longitud, no puede utilizarse una serie de objeto grande, un valor LONG VARCHAR ni un valor LONG VARGRAPHIC para representar un valor de fecha y hora (SQLSTATE 42884).

Cuando se utiliza una representación de serie válida de un valor de fecha y hora en una operación con un valor de fecha y hora interno, la representación de serie se convierte al formato interno del valor de fecha, hora o indicación de fecha y hora antes de realizar la operación.

Las series de fecha, hora e indicación de fecha y hora sólo deben contener caracteres y dígitos.

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

Los formatos válidos para las series se indican en la tabla siguiente. Cada formato se identifica mediante el nombre y la abreviatura asociada.

Tabla 5. Formatos para las representaciones de serie de fechas

Nombre del formato	Abreviatura	Formato de fecha	Ejemplo
International Standards Organization	ISO	aaaa-mm-dd	1991-10-27
Estándar IBM USA	USA	mm/dd/aaaa	10/27/1991
Estándar IBM European	EUR	dd.mm.aaaa	27.10.1991
Era Japanese Industrial Standard Christian	JIS	aaaa-mm-dd	1991-10-27
Definido-sitio	LOC	Depende del código territorial de la aplicación	—

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

Los formatos válidos para las series de horas se indican en la tabla siguiente. Cada formato se identifica mediante el nombre y la abreviatura asociada.

Tabla 6. Formatos para representaciones de serie de horas

Nombre del formato	Abreviatura	Formato de la hora	Ejemplo
International Standards Organization[2]	ISO	hh.mm.ss	13.30.05
Estándar IBM USA	USA	hh:mm AM o PM	1:30 PM
Estándar IBM European	EUR	hh.mm.ss	13.30.05
Era Japanese Industrial Standard Christian	JIS	hh:mm:ss	13:30:05
Definido-sitio	LOC	Depende del código territorial de la aplicación	—

Notas:

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

   12:01 AM a 12:59 AM corresponde a 00:01:00 a 00:59:00.

   01:00 AM a 11:59 AM corresponde a 01:00:00 a 11:59:00.

   12:00 PM (mediodía) a 11:59 PM corresponde a 12:00:00 a 23:59:00.

   12:00 AM (medianoche) corresponde a 24:00:00 y 00:00 AM (medianoche) corresponde a 00:00:00.

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

Las sentencias de SQL también dan soporte a la representación de serie ODBC de una indicación de fecha y hora, pero sólo como un valor de entrada. La representación de serie ODBC de una indicación de fecha y hora tiene el formato *aaaa-mm-dd hh:mm:ss.nnnnnn*.

Valores DATALINK

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

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

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

- HTTP
- FILE
- UNC

Las demás partes del URL son:

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

comentario
Hasta 200 bytes de información descriptiva, *incluido* el atributo de ubicación de los datos. Está pensado para los usos específicos de una aplicación, como, por ejemplo, una identificación alternativa o más detallada de la ubicación de los datos.

Los caracteres en blanco iniciales y de cola se eliminan durante el análisis de los atributos de ubicación de datos en forma de URL. Además, los nombres de esquema ('http', 'file', 'unc') y de lenguaje principal no son sensibles a las mayúsculas/minúsculas y siempre se guardan en mayúsculas en la base de datos. Cuando se recupera un valor DATALINK de una base de datos, se intercala un símbolo de accesos dentro del atributo de URL si la columna DATALINK está definida con READ PERMISSION DB o WRITE PERMISSION ADMIN. El símbolo se genera dinámicamente y no es una parte permanente del valor DATALINK almacenado en la base de datos.

Un valor DATALINK puede tener solamente un atributo de comentario y un atributo vacío de ubicación de datos. Incluso es posible que un valor de este tipo se almacene en una columna, pero, naturalmente, no se enlazará ningún archivo a esta columna. La longitud total del comentario y el atributo de ubicación de datos de un valor DATALINK está actualmente limitado a 200 bytes.

Es importante distinguir entre las referencias DATALINK a archivos y las variables de referencia a archivos LOB. La semejanza es que ambas contienen una representación de un archivo. No obstante:

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

Utilice las funciones escalares incorporadas para crear un valor DATALINK (DLVALUE, DLNEWCOPY, DLPREVIOUSCOPY y DLREPLACECONTENT) y para extraer los valores encapsulados de un valor DATALINK (DLCOMMENT,

DLLINKTYPE, DLURLCOMPLETE, DLURLPATH, DLURLPATHONLY, DLURLSCHEME, DLURLSERVER, DLURLCOMPLETEONLY, DLURLCOMPLETEWRITE y DLURLPATHWRITE).

Información relacionada:

- "Identificadores" en la página 64
- Apéndice O, "Especificaciones de formato de Backus-Naur (BNF) para los enlaces de datos", en la página 809

Valores XML

El tipo de datos XML es una representación interna de XML y sólo puede utilizarse como entrada a funciones que acepten este datos como entrada. XML es un tipo de datos transitorio que no puede almacenarse en la base de datos ni devolverse a una aplicación.

Entre los valores válidos para el tipo de datos XML se incluyen los siguientes:

- Un elemento
- Un bosque de elementos
- El contenido textual de un elemento
- Un valor XML vacío

Actualmente, la única operación que se permite consiste en serializar (utilizando la función XML2CLOB) el valor XML en una serie que se almacena como un valor CLOB.

Tipos definidos por el usuario

Existen tres tipos de datos definidos por el usuario:

- Tipo diferenciado
- Tipo estructurado
- Tipo de referencia

Cada uno de ellos se describe en los apartados siguientes.

Tipos diferenciados

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

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

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

Aunque AUDIO tenga la misma representación que el tipo de datos interno BLOB, se considera un tipo independiente; esto permite la creación de funciones escritas especialmente para AUDIO y asegura que dichas funciones no se aplicarán a valores de ningún otro tipo de datos (imágenes, texto, etc.)

Los tipos diferenciados tienen identificadores calificados. Si no se utiliza el nombre de esquema para calificar el nombre del tipo diferenciado cuando se emplea en sentencias que no son CREATE DISTINCT TYPE, DROP DISTINCT TYPE o COMMENT ON DISTINCT TYPE, en la vía de acceso de SQL se busca por orden el primer esquema con un tipo diferenciado que coincida.

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

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

Sin embargo, se puede especificar explícitamente que ciertas funciones y ciertos operadores del tipo fuente se apliquen al tipo diferenciado. Esto puede hacerse creando funciones definidas por el usuario que se deriven de funciones definidas en el tipo fuente del tipo diferenciado. Los operadores de comparación se generan automáticamente para los tipos diferenciados definidos por el usuario, excepto los que utilicen LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB o DATALINK como tipo fuente. Además, se generan funciones para dar soporte a la conversión del tipo fuente al tipo diferenciado y del tipo diferenciado al tipo fuente.

Tipos estructurados

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

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

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

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

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

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

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

Un tipo no se puede descartar si ciertos otros objetos utilizan el tipo, ya sea directa o indirectamente. Por ejemplo, no se puede descartar un tipo si una columna de una tabla o vista hace uso directa o indirectamente del tipo.

Tipos de referencia

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

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

Información relacionada:

- "Sentencia DROP" en la publicación *Consulta de SQL, Volumen 2*
- "CURRENT PATH" en la página 152
- "Series de caracteres" en la página 90
- "Asignaciones y comparaciones" en la página 110

Promoción de tipos de datos

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

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

- Para realizar la resolución de la función
- Para convertir tipos definidos por el usuario
- Para asignar tipos definidos por el usuario a tipos de datos internos

La Tabla 7 muestra la lista de prioridad (por orden) para cada tipo de datos y se puede utilizar para determinar los tipos de datos a los que se puede promover un tipo de datos determinado. La tabla muestra que la mejor elección siempre es el mismo tipo de datos en lugar de elegir la promoción a otro tipo de datos.

Tabla 7. Tabla de prioridades de tipos de datos

Tipo de datos	Lista de prioridad de tipos de datos (por orden de mejor a peor)
CHAR	CHAR, VARCHAR, LONG VARCHAR, CLOB
VARCHAR	VARCHAR, LONG VARCHAR, CLOB
LONG VARCHAR	LONG VARCHAR, CLOB
GRAPHIC	GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB
VARGRAPHIC	VARGRAPHIC, LONG VARGRAPHIC, DBCLOB
LONG VARGRAPHIC	LONG VARGRAPHIC, DBCLOB
BLOB	BLOB
CLOB	CLOB
DBCLOB	DBCLOB
SMALLINT	SMALLINT, INTEGER, BIGINT, decimal, real, double
INTEGER	INTEGER, BIGINT, decimal, real, double
BIGINT	BIGINT, decimal, real, double
decimal	decimal, real, double
real	real, double
double	double
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
DATALINK	DATALINK
udt	udt (mismo nombre) o un supertipo de udt
REF(T)	REF(S) (en el caso de que S sea un supertipo de T)

Tabla 7. Tabla de prioridades de tipos de datos (continuación)

Tipo de datos	Lista de prioridad de tipos de datos (por orden de mejor a peor)

Notas:

1. Los tipos en minúsculas anteriores se definen de la siguiente manera:
   - decimal = DECIMAL(p,s) o NUMERIC(p,s)
   - real = REAL o FLOAT(*n*), donde *n* not es mayor que 24
   - double = DOUBLE, DOUBLE-PRECISION, FLOAT o FLOAT(*n*), donde *n* es mayor que 24
   - udt = un tipo definido por el usuario

   Los sinónimos, más cortos o más lagos, de los tipos de datos listados se consideran iguales a la forma listada.
2. Para una base de datos Unicode, los siguientes se consideran tipos de datos equivalentes:
   - CHAR y GRAPHIC
   - VARCHAR y VARGRAPHIC
   - LONG VARCHAR y LONG VARGRAPHIC
   - CLOB y DBCLOB

Información relacionada:

- "Funciones" en la página 164
- "Conversiones entre tipos de datos" en la página 107
- "Asignaciones y comparaciones" en la página 110

Conversiones entre tipos de datos

En muchas ocasiones, un valor con un tipo de datos determinado necesita *convertirse* a un tipo de datos diferente o al mismo tipo de datos con una longitud, precisión o escala diferentes. La promoción del tipo de datos es un ejemplo en el que la promoción de un tipo de datos a otro tipo de datos necesita que el valor se convierta al nuevo tipo de datos. Un tipo de datos que se puede convertir a otro tipo de datos es *convertible* de un tipo de datos fuente al tipo de datos de destino.

La conversión entre tipos de datos puede realizarse explícitamente utilizando la especificación CAST pero también puede efectuarse implícitamente durante las asignaciones que implican tipos definidos por el usuario. Además, al crear funciones definidas por el usuario derivadas, los tipos de datos de los parámetros de la función fuente deben poder convertirse a los tipos de datos de la función que se está creando.

La Tabla 8 en la página 109 muestra las conversiones permitidas entre tipos de datos internos. La primera columna representa el tipo de datos del operando cast (tipo de datos fuente) y los tipos de datos en la parte superior representan el tipo de datos de destino de la especificación CAST.

En una base de datos Unicode, si se produce un truncamiento al convertir una serie de caracteres o gráfica a otro tipo de datos, se devuelve una advertencia si se trunca algún carácter que no sea un blanco. Este comportamiento de truncamiento es distinto de la asignación de series de caracteres o gráficas a un destino cuando se produce un error si se trunca algún carácter que no es un blanco.

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

- Conversión de un tipo diferenciado *DT* a su tipo de datos fuente *S*
- Conversión del tipo de datos fuente *S* de un tipo diferenciado *DT* al tipo diferenciado *DT*
- Conversión del tipo diferenciado *DT* al mismo tipo diferenciado *DT*
- Conversión de un tipo de datos *A* a un tipo diferenciado *DT* donde *A* se puede promocionar al tipo de datos fuente *S* del tipo diferenciado *DT*
- Conversión de INTEGER a un tipo diferenciado *DT* con un tipo de datos fuente SMALLINT
- Conversión de DOUBLE a un tipo diferenciado *DT* con un tipo de datos fuente REAL
- Conversión de VARCHAR a un tipo diferenciado *DT* con un tipo de datos fuente CHAR
- Conversión de VARGRAPHIC a un tipo diferenciado *DT* con un tipo de datos fuente GRAPHIC
- Para una base de datos Unicode, conversión de VARCHAR o VARGRAPHIC a un tipo diferenciado *DT* con un tipo de datos fuente CHAR o GRAPHIC.

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

No es posible convertir un valor de tipo estructurado en algo diferente. Un tipo estructurado *ST* no necesita convertirse a uno de sus supertipos, porque todos los

métodos de los supertipos de *ST* son aplicables a *ST*. Si la operación deseada sólo es aplicable a un subtipo de *ST*, utilice la expresión de tratamiento de subtipos para tratar *ST* como uno de sus subtipos.

Cuando un tipo de datos definido por el usuario e implicado en una conversión no está calificado por un nombre de esquema, se utiliza la *vía de acceso de SQL* para buscar el primer esquema que incluya el tipo de datos definido por el usuario con este nombre.

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

- conversión de un tipo de referencia *RT* en su tipo de datos de representación *S*
- conversión del tipo de datos de representación *S* de un tipo de referencia *RT* en el tipo de referencia *RT*
- Conversión de un tipo de referencia *RT* con un tipo de destino *T* a un tipo de referencia *RS* con un tipo de destino *S* donde *S* es un supertipo de *T*.
- Conversión de un tipo de datos *A* en un tipo de referencia *RT* donde *A* se puede promocionar al tipo de datos de representación *S* del tipo de referencia *RT*.

Cuando el tipo de destino de un tipo de datos de referencia implicado en una conversión no está calificado por un nombre de esquema, se utiliza la *vía de acceso de SQL* para buscar el primer esquema que incluya el tipo de datos definido por el usuario con este nombre.

Tabla 8. Conversiones soportadas entre tipos de datos internos

Tipo de datos de destino → / Tipo de datos fuente ↓	SMALLINT	INTEGER	BIGINT	DECIMAL	REAL	DOUBLE	CHAR	VARCHAR	LONG VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	LONG VARGRAPHIC	DBCLOB	DATE	TIME	TIMESTAMP	BLOB
SMALLINT	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
INTEGER	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
BIGINT	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
DECIMAL	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
REAL	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-
DOUBLE	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-
CHAR	Y	Y	Y	Y	-	-	Y	Y	Y	Y	Y[1]	Y[1]	-	-	Y	Y	Y	Y
VARCHAR	Y	Y	Y	Y	-	-	Y	Y	Y	Y	Y[1]	Y[1]	-	-	Y	Y	Y	Y
LONG VARCHAR	-	-	-	-	-	-	Y	Y	Y	Y	-	-	Y[1]	Y[1]	-	-	-	Y
CLOB	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	Y[1]	-	-	-	Y
GRAPHIC	-	-	-	-	-	-	Y[1]	Y[1]	-	-	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y
VARGRAPHIC	-	-	-	-	-	-	Y[1]	Y[1]	-	-	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y
LONG VARGRAPHIC	-	-	-	-	-	-	-	-	Y[1]	Y[1]	Y	Y	Y	Y	-	-	-	Y
DBCLOB	-	-	-	-	-	-	-	-	-	Y[1]	Y	Y	Y	Y	-	-	-	Y
DATE	-	Y	Y	Y	-	-	Y	Y	-	-	Y[1]	Y[1]	-	-	Y	-	-	-
TIME	-	Y	Y	Y	-	-	Y	Y	-	-	Y[1]	Y[1]	-	-	-	Y	-	-
TIMESTAMP	-	-	Y	Y	-	-	Y	Y	-	-	Y[1]	Y[1]	-	-	Y	Y	Y	-
BLOB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y

Notas

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

[1] Conversión sólo soportada para bases de datos Unicode.

Información relacionada:

- "Expresiones" en la página 181
- "Sentencia CREATE FUNCTION" en la publicación *Consulta de SQL, Volumen 2*
- "CURRENT PATH" en la página 152
- "Promoción de tipos de datos" en la página 105
- "Asignaciones y comparaciones" en la página 110

Asignaciones y comparaciones

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

Una regla básica para las dos operaciones es que el tipo de datos de los operandos implicados debe ser compatible. La regla de compatibilidad también se aplica a las operaciones de conjuntos.

Otra regla básica para las operaciones de asignación es que no pueda asignarse un valor nulo a una columna que no pueda contener valores nulos, ni a una variable del lenguaje principal que no tenga una variable indicadora asociada.

Sólo se da soporte a las asignaciones y comparaciones que implican datos tanto de caracteres como gráficos cuando una de las series es un literal.

La matriz de compatibilidad a continuación que muestra las compatibilidades de tipos de datos para operaciones de asignación y comparación.

Tabla 9. Compatibilidad de tipos de datos para asignaciones y comparaciones

Operandos	Entero binario	Número decimal	Coma flotante	Serie de caracteres	Serie gráfica	Fecha	Hora	Indicación de fecha y hora	Serie binaria	UDT
Entero binario	Sí	Sí	Sí	No	No	No	No	No	No	2
Número decimal	Sí	Sí	Sí	No	No	No	No	No	No	2
Coma flotante	Sí	Sí	Sí	No	No	No	No	No	No	2
Serie de caracteres	No	No	No	Sí	Sí 6,7	1	1	1	No 3	2
Serie gráfica	No	No	No	Sí 6,7	Sí	1	1	1	No	2
Fecha	No	No	No	1	1	Sí	No	No	No	2
Hora	No	No	No	1	1	No	Sí	No	No	2
Indicación de fecha y hora	No	No	No	1	1	No	No	Sí	No	2
Serie binaria	No	No	No	No 3	No	No	No	No	Sí	2
UDT	2	2	2	2	2	2	2	2	2	Sí

Tabla 9. Compatibilidad de tipos de datos para asignaciones y comparaciones (continuación)

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

[1] La compatibilidad de los valores de indicación de fecha y hora y de las series está limitada a la asignación y la comparación:

- Los valores de fecha y hora se pueden asignar a las columnas de series y a las variables de series.
- Una representación de serie válida de una fecha se puede asignar a una columna de fecha o comparar con una fecha.
- Una representación de serie válida de una hora se puede asignar a una columna de hora o comparar con una hora.
- Una representación de serie válida de una indicación de fecha y hora se puede asignar a una columna de indicación de fecha y hora o comparar con una indicación de fecha y hora.

(El soporte a series gráficas sólo está disponible para bases de datos Unicode.)

[2] Un valor de tipo diferenciado definido por el usuario sólo se puede comparar con un valor definido con el mismo tipo diferenciado definido por el usuario. En general, se da soporte a las asignaciones entre un valor de tipo diferenciado y su tipo de datos fuente. Un tipo estructurado definido por el usuario no es comparable y sólo se puede asignar a un operando del mismo tipo estructurado o a uno de sus subtipos. Para obtener más información, vea "Asignación de tipos definidos por el usuario" en la página 119.

[3] Observe que esto significa que las series de caracteres definidas con el atributo FOR BIT DATA tampoco son compatibles con las series binarias.

[4] Un operando DATALINK sólo puede asignarse a otro operando DATALINK. El valor DATALINK sólo puede asignarse a una columna si la columna está definida con NO LINK CONTROL o el archivo existe y todavía no está bajo el control del enlace del archivo.

[5] Para obtener información sobre la asignación y comparación de tipos de referencia, consulte "Asignación de tipos de referencia" en la página 119 y "Comparaciones de tipos de referencia" en la página 124.

[6] Sólo soportado para bases de datos Unicode.

[7] Las series de datos de bits y gráficas no son compatibles.

Asignaciones numéricas

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

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

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

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

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

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

Asignaciones de series

Existen dos tipos de asignaciones:

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

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

Asignación de almacenamiento: La regla básica es que la longitud de la serie asignada a una columna o a un parámetro de rutina no debe ser mayor que el atributo de longitud de la columna o del parámetro de rutina. Si la longitud de la serie es mayor que el atributo de longitud de la columna o del parámetro de rutina, se pueden producir las siguientes situaciones:

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

Si se asigna una serie a una columna de longitud fija y la longitud de la serie es menor que el atributo de longitud del destino, la serie se rellena por la derecha con el número necesario de blancos de un solo byte, de doble byte o UCS-2. El carácter de relleno siempre es el blanco, incluso para las columnas definidas con el atributo FOR BIT DATA. (UCS-2 define varios caracteres SPACE con distintas propiedades. Para una base de datos Unicode, el gestor de bases de datos siempre utilizará ASCII SPACE en la posición x'0020' como blanco UCS-2. Para una base de datos EUC, se utiliza IDEOGRAPHIC SPACE en la posición x'3000' para rellenar las series GRAPHIC.)

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

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

Si se asigna una serie de caracteres a una variable de longitud fija y la longitud de la serie es menor que el atributo de longitud del destino, la serie se rellena por la derecha con el número necesario de blancos de un solo byte, de doble byte o UCS-2. El carácter de relleno siempre es un blanco, incluso para las series definidas con el atributo FOR BIT DATA. (UCS-2 define varios caracteres SPACE con distintas propiedades. Para una base de datos Unicode, el gestor de bases de datos siempre utilizará ASCII SPACE en la posición x'0020' como blanco UCS-2. Para una base de datos EUC, se utiliza IDEOGRAPHIC SPACE en la posición x'3000' para rellenar las series GRAPHIC.)

La asignación de recuperación de las variables del lenguaje principal terminadas en nulo en C se maneja en base a las opciones especificadas con los mandatos PREP o BIND.

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

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

Para las bases de datos Unicode, las series de caracteres pueden asignarse a una columna gráfica y las series gráficas pueden asignarse a una columna de caracteres.

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

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

Consideraciones de DBCS para la asignación de las series gráficas: Las asignaciones de series gráficas se procesan de manera análoga a la de las series de caracteres. Para las bases de datos que no son Unicode, los tipos de datos de serie gráfica sólo son compatibles con otros tipos de datos de serie gráfica y nunca con tipos de datos numéricos, de serie de caracteres o de indicación de fecha y hora. Para las bases de Unicode, los tipos de datos de serie gráfica son compatibles con tipos de datos de serie de caracteres.

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

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

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

La asignación de recuperación de las variables del lenguaje principal terminadas en nulo en C (declaradas utilizando wchar_t) se maneja sobre la base de las opciones especificadas con los mandatos PREP o BIND.

Asignaciones de fecha y hora

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

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

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

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

  Si se utilizan los formatos ISO o JIS, y la longitud de la variable del lenguaje principal es menor que 8 caracteres, la parte correspondiente a los segundos de

la hora se omite del resultado y se asigna a la variable indicadora, si se proporciona. El campo SQLWARN1 de la SQLCA se establece de manera que indica la omisión.

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

Asignaciones de DATALINK

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

Si la aplicación proporciona la misma ubicación de datos que la que ya existe en la columna, se retiene el enlace. Hay varias razones por las que puede producirse este hecho:

- Se cambia el comentario.
- Si la tabla se coloca en el estado de reconciliación de Datalink no posible (DRNP), pueden restablecerse los enlaces en la tabla al proporcionarse unos atributos de enlace idénticos a los de la columna.
- Si la columna se define con WRITE PERMISSION ADMIN y se cambia el contenido del archivo, puede establecerse una versión nueva del enlace proporcionando un valor de DATALINK construido utilizando la función DLURLNEWCOPY con la misma ubicación de los datos.
- Si la columna se define con WRITE PERMISSION ADMIN y se cambia el contenido del archivo pero el cambio debe restituirse, puede restituirse la versión existente del enlace proporcionando un valor de DATALINK construido utilizando la función DLURLPREVIOUSCOPY con la misma ubicación de los datos.
- El contenido del archivo al que se hace referencia se sustituye por otro archivo especificado en la función escalar DLURLREPLACECONTENT.

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

- Se puede utilizar la función escalar DLVALUE para crear un nuevo valor DATALINK y asignarlo a una columna. A no ser que el valor sólo contenga un comentario o que el URL sea exactamente el mismo, la acción de asignación enlazará el archivo.
- Se puede crear un valor DATALINK en un parámetro de CLI con la función SQLBuildDataLink de CLI. A continuación, este valor puede asignarse a una columna. A no ser que el valor sólo contenga un comentario o que el URL sea exactamente el mismo, la acción de asignación enlazará el archivo.
- Se puede utilizar la función escalar DLURLNEWCOPY para construir un valor de DATALINK y asignarlo a una columna. La ubicación de los datos a la que hace referencia el valor de DATALINK construido debe ser la misma que la que ya existe en la columna. Con la asignación mediante una sentencia UPDATE se restablece el enlace con el archivo. Se tomará una copia de seguridad del archivo si la columna se define con RECOVERY YES. Este tipo de asignación se utiliza

para comunicar a la base de datos que el archivo se ha actualizado. Así se informa a la base de datos de la existencia del archivo nuevo y ésta restablecerá un enlace nuevo con el archivo.

- Se puede utilizar la función escalar DLURLPREVIOUSCOPY para construir un valor de DATALINK y asignarlo a una columna. La ubicación de los datos a la que hace referencia el valor de DATALINK construido debe ser la misma que la que ya existe en la columna. Con la asignación mediante una sentencia UPDATE se restituye el enlace. También se restaurará el archivo en la versión anterior si la columna está definida con RECOVERY YES. Este tipo de asignación se utiliza para restituir los cambios realizados en el archivo desde la versión anterior confirmada.
- Se puede utilizar la función escalar DLURLREPLACECONTENT para crear un nuevo valor DATALINK y asignarlo a una columna. Con la asignación no sólo se enlazará el archivo, sino que también se sustituirá el contenido con otro archivo especificado en la función escalar DLURLREPLACECONTENT.

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

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

  Tenga en cuenta que el tamaño de un resultado de función o parámetro de URL es el mismo en la entrada y en la salida y que dicho tamaño está limitado por la longitud de la columna DATALINK. Sin embargo, en algunos casos el valor del URL se devuelve con un símbolo de accesos adjunto. En las situaciones en que esto sea posible, la ubicación de la salida debe tener espacio de almacenamiento suficiente para el símbolo de accesos y la longitud de la columna DATALINK. En consecuencia, la longitud real del comentario y del URL (en su formato totalmente ampliado) que se proporciona en la entrada debe restringirse para adaptarse al espacio de almacenamiento de la salida. Si se sobrepasa la longitud restringida, surge este error.
- La ubicación de los datos de entrada no contiene un símbolo de escritura válido (código de razón 32).

  La asignación requiere que en la ubicación de los datos exista un símbolo de escritura válido. Este requisito sólo se aplica cuando la columna está definida con WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE y el valor de DATALINK está construido mediante la función escalar DLURLNEWCOPY o DLURLPREVIOUSCOPY. Por otra parte, un usuario tiene la opción de proporcionar un símbolo de escritura para una columna DATALINK definida con WRITE PERMISSION ADMIN NOT REQUIRING TOKEN FOR UPDATE. Sin embargo, si el símbolo no es válido, se produce el mismo error.

  Este error también puede producirse al construir un valor de DATALINK si se utiliza la función escalar DLURLNEWCOPY o DLURLPREVIOUSCOPY con el valor '1' especificado en el segundo argumento pero el valor no contiene un símbolo de escritura válido.
- El valor de DATALINK construido por la función escalar DLURLPREVIOUSCOPY sólo puede asignarse a una columna DATALINK definida con WRITE PERMISSION ADMIN y RECOVERY YES (código de razón 33).

- El valor de DATALINK construido por la función escalar DLURLNEWCOPY o DLURLPREVIOUSCOPY no coincide con el valor que ya existe en la columna (código de razón 34).
- El valor de DATALINK construido por la función escalar DLURLNEWCOPY o DLURLPREVIOUSCOPY no puede utilizarse en una sentencia INSERT para asignar un valor nuevo (código de razón 35).
- El valor de DATALINK construido por la función escalar DLURLNEWCOPY no puede asignarse a una columna DATALINK definida con WRITE PERMISSION BLOCKED (código de razón 39).
- El mismo valor de DATALINK construido por la función escalar DLURLNEWCOPY o DLURLPREVIOUSCOPY no puede asignarse varias veces en la misma transacción (código de razón 41).
- El valor de DATALINK construido por la función escalar DLURLREPLACECONTENT sólo puede asignarse a una columna DATALINK definida con NO LINK CONTROL si el segundo argumento es una serie vacía o un valor nulo (código de razón 42).
- La operación de desenlace del archivo de sustitución especificado en la función escalar DLREPLACECONTENT no se ha confirmado (código de razón 43).
- El archivo de sustitución especificado en la función escalar DLREPLACECONTENT se está utilizando en otro proceso de sustitución (código de razón 44).
- El archivo que hace referencia a DATALINK se está utilizando como archivo de sustitución en otra operación (código de razón 45).
- El formato del archivo de sustitución especificado en la función escalar DLREPLACECONTENT no es válido (código de razón 46).
- El archivo de sustitución especificado en la función escalar DLREPLACECONTENT no puede ser un directorio ni un enlace simbólico (código de razón 47).
- El archivo de sustitución especificado en la función escalar DLREPLACECONTENT se está enlazando con una base de datos (código de razón 48).
- El gestor de archivos Data Links no puede encontrar el archivo de sustitución especificado en la función escalar DLREPLACECONTENT (código de razón 49).
- El valor de DATALINK construido por la función escalar DLURLNEWCOPY con un símbolo de escritura contenido en el valor de ubicación de los datos sólo puede asignarse a una columna DATALINK con WRITE PERMISSION ADMIN (código de razón 50).

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

- El servidor de archivos no está disponible actualmente (SQLSTATE 57050).
- El archivo no existe (SQLSTATE 428D1, código de razón 24).
- El archivo ya está enlazado con otra columna (SQLSTATE 428D1, código de razón 25).

  Tenga en cuenta que surgirá este error aunque el enlace sea con una base de datos diferente.
- No se puede acceder al archivo de referencia para el enlace (código de razón 26).
- El símbolo de escritura intercalado en la ubicación de los datos no coincide con el símbolo de escritura utilizado para abrir el archivo (SQLSTATE 428D1, código de razón 36).

- El archivo al que hace referencia el valor de DATALINK está en estado de actualización en proceso (SQLSTATE 428D1, código de razón 37).
- La copia archivada anterior del archivo al que hace referencia el valor de DATALINK no está disponible (SQLSTATE 428D1, código de razón 40).

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

- El servidor de archivos no está disponible actualmente (SQLSTATE 57050).
- El archivo con control de integridad referencial no presenta un estado correcto según Data Links File Manager (SQLSTATE 58004).
- El archivo al que hace referencia el valor de DATALINK está en estado de actualización en proceso (SQLSTATE 428D1, código de razón 37).

Si al recuperar un valor de DATALINK para acceso de escritura (utilizando la función escalar DLURLCOMPLETEWRITE o DLURLPATHWRITE), la columna DATALINK se define con WRITE PERMISSION ADMIN, se comprueba el privilegio de acceso a directorios en el servidor de archivos (gestor de archivos DataLink). El usuario que envía la consulta debe tener la autorización para grabar en los archivos bajo dicho directorio antes de que se genere un símbolo de escritura y se intercale en el valor devuelto por DATALINK. Si el usuario no tiene autorización para grabar, no se generará ningún símbolo y la sentencia SELECT será insatisfactoria (SQLSTATE 42511, código de razón 1).

Pueden asignarse partes de un valor de DATALINK a variables del lenguaje principal mediante la aplicación de funciones escalares como, por ejemplo, DLLINKTYPE o DLURLPATH).

Tenga en cuenta que, normalmente, no se intenta acceder al servidor de archivos en el momento de la recuperación. Por lo tanto, es posible que fallen los intentos subsiguientes de acceso al servidor de archivos mediante mandatos de sistema de archivos. Puede que sea necesario acceder al servidor de archivos para determinar el nombre de prefijo asociado con una vía de acceso. Se puede cambiar en el servidor de archivos cuando se mueva el punto de montaje de un sistema de archivos. El primer acceso a un archivo de un servidor causará que se recuperen los valores necesarios del servidor de archivos y se coloquen en la antememoria del servidor de bases de datos para la subsiguiente recuperación de valores DATALINK de este servidor de archivos. Se devuelve un error si no se puede acceder al servidor de archivos (SQLSTATE 57050).

Si se utilizan las funciones escalares DLURLCOMPLETEWRITE o DLURLPATHWRITE para recuperar un valor de DATALINK, es posible que sea necesario acceder al servidor de archivos para determinar el privilegio de acceso a directorios que posee un usuario sobre una vía de acceso. Se devuelve un error si no se puede acceder al servidor de archivos (SQLSTATE 57050).

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

Asignación de tipos definidos por el usuario

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

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

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

Si el destino de la asignación es una columna basada en un tipo diferenciado, el tipo de datos fuente debe ser convertible al tipo de datos de destino.

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

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

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

Asignación de tipos de referencia

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

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

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

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

Comparaciones numéricas

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

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

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

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

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

Comparaciones de series

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

Cuando se comparan series de caracteres de longitudes desiguales y el orden de clasificación especificado no es de tipo UCA (Algoritmo de clasificación Unicode), la comparación se realiza utilizando una copia lógica de la serie más corta que se ha rellenado por la derecha con suficientes blancos de un solo byte para extender su longitud a la de la serie más larga. Esta extensión lógica se realiza para todas las series de caracteres, incluidas las que tienen el distintivo FOR BIT DATA. Si el orden de clasificación es de tipo UCA, la serie más corta no se rellena por la derecha, porque si se rellena con algún carácter el resultado de la comparación podría verse afectado. Las comparaciones utilizando UCA se realizan en uno o más pesos de cada carácter de las series y no en los elementos de código. En el Estándar técnico de Unicode número 10, disponible en el sitio WebUnicode Consortium hallará detalles sobre el tipo UCA. La información siguiente pertenece a los órdenes de clasificación que no son de tipo UCA.

Las series de caracteres (excepto las series de caracteres con el distintivo FOR BIT DATA) se comparan de acuerdo con el orden de clasificación especificado al crear la base de datos. Por ejemplo, el orden de clasificación por omisión proporcionado por el gestor de bases de datos puede dar el mismo peso a la versión en minúsculas y en mayúsculas del mismo carácter. El gestor de bases de datos realiza una comparación en dos pasos para asegurarse de que sólo se consideran iguales las series idénticas. En el primer paso, las series se comparan de acuerdo con el orden de clasificación de la base de datos. Si los pesos de los caracteres de las series son iguales, se realiza un segundo paso de "desempate" para comparar las series en base a sus valores de elemento de código real.

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

No se da soporte a las series largas ni a las series LOB las operaciones de comparación que utilizan operadores de comparación básicos (=, <>, <, >, <= y >=). Están soportadas en comparaciones que utilizan el predicado LIKE y la función POSSTR.

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

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

entonces lo siguiente será válido:

```
WHERE VARCHAR(MY_SHORT_CLOB) > VARCHAR(SUBSTR(MY_LONG_VAR,1,300))
```

Ejemplos:

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

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

```
'a' < 'A' < 'á' < 'Á'
```

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

```
D4 < D3 < D2 < D1
```

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

```
 'A' < 'a' < 'Á' < 'á'
```

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

```
D1 < D2 < D3 < D4
```

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

```
'A' < 'a' < 'Á' < 'á'
```

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

```
D1 < D2 < D3 < D4
```

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

```
D1 < D2 < D3 < D4
```

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

Reglas de conversión para la comparación: Cuando se comparan dos series, primero una de ellas se convierte, si es necesario, al esquema de codificación y a la página de códigos de la otra serie.

Clasificación de los resultados: Los resultados que necesitan clasificarse se ordenan en base a las reglas de comparación de series que se tratan en "Comparaciones de series" en la página 120. La comparación se efectúa en el servidor de bases de datos. Al devolver los resultados a la aplicación cliente, se puede llevar a cabo una conversión de la página de códigos. Esta conversión de la pagina de códigos subsiguiente no afecta al orden del conjunto resultante determinado por el servidor.

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

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

Ejemplo:

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

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

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

y

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

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

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

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

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

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

y

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

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

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

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

Comparaciones de fecha y hora

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

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

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

Ejemplo:

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

Comparaciones de tipos definidos por el usuario

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

Ejemplo:

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

```
CREATE DISTINCT TYPE YOUTH AS INTEGER WITH COMPARISONS

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

La comparación siguiente es válida:

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > HIGH_SCHOOL_LEVEL
```

La comparación siguiente no es válida:

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > ATTENDEE_NUMBER
```

Sin embargo, AGE se puede comparar con ATTENDEE_NUMBER utilizando una función o una especificación CAST para convertir entre el tipo diferenciado y el tipo fuente. Todas las comparaciones siguientes son válidas:

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE INTEGER(AGE) > ATTENDEE_NUMBER

SELECT * FROM CAMP_DB2_ROSTER
     WHERE CAST( AGE AS INTEGER) > ATTENDEE_NUMBER

SELECT * FROM CAMP_DB2_ROSTER
     WHERE AGE > YOUTH(ATTENDEE_NUMBER)

SELECT * FROM CAMP_DB2_ROSTER
     WHERE AGE > CAST(ATTENDEE_NUMBER AS YOUTH)
```

Los valores con un tipo estructurado definido por el usuario no se pueden comparar con ningún otro valor (se pueden utilizar los predicados NULL y TYPE).

Comparaciones de tipos de referencia

Los valores de un tipo de referencia sólo pueden compararse si sus tipos de destino tienen un supertipo común. Sólo se encontrará la función de comparación adecuada si el nombre de esquema del supertipo común se incluye en la vía de acceso de la función. Se realiza la comparación si se utiliza el tipo de representación de los tipos de referencia. El ámbito de la referencia no se tiene en cuenta en la comparación.

Información relacionada:

- “Identificadores” en la página 64
- “Predicado LIKE” en la página 231
- “POSSTR” en la página 403
- “Valores de fecha y hora” en la página 96
- “Conversiones entre tipos de datos” en la página 107
- “Reglas para los tipos de datos del resultado” en la página 125
- “Reglas para la conversión de series” en la página 129

Reglas para los tipos de datos del resultado

Los tipos de datos de un resultado los determinan las reglas que se aplican a los operandos de una operación. Esta sección explica dichas reglas.

Estas reglas se aplican a:

- Las columnas correspondientes en selecciones completas de operaciones de conjuntos (UNION, INTERSECT y EXCEPT)
- Expresiones resultantes de una expresión CASE
- Los argumentos de la función escalar COALESCE (o VALUE)
- Los valores de expresiones de la lista de entrada de un predicado IN
- Las expresiones correspondientes de una cláusula VALUES de múltiples filas.

Estas reglas se aplican, sujetas a otras restricciones, sobre series largas para las distintas operaciones.

A continuación encontrará las reglas que se refieren a los distintos tipos de datos. En algunos casos, se utiliza una tabla para mostrar los posibles tipos de datos resultantes.

Estas tablas identifican el tipo de datos resultante, incluida la longitud aplicable o precisión y la escala. El tipo resultante se determina teniendo en cuenta los operandos. Si hay más de un par de operandos, empiece considerando el primer par. Esto da un tipo resultante que es el que se examina con el siguiente operando para determinar el siguiente tipo resultante, etcétera. El último tipo resultante intermedio y el último operando determinan el tipo resultante para la operación. El proceso de operaciones se realiza de izquierda a derecha, por lo tanto los tipos del resultado intermedios son importantes cuando se repiten operaciones. Por ejemplo, examinemos una situación que implique:

```
CHAR(2) UNION CHAR(4) UNION VARCHAR(3)
```

El primer par da como resultado un tipo CHAR(4). Los valores del resultado siempre tienen 4 caracteres. El tipo resultante final es VARCHAR(4). Los valores del resultado de la primera operación UNION siempre tendrán una longitud de 4.

Series de caracteres

Las series de caracteres son compatibles con otras series de caracteres. Las series de caracteres incluyen los tipos CHAR, VARCHAR, LONG VARCHAR y CLOB.

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
CHAR(x)	CHAR(y)	CHAR(z) donde z = max(x,y)
CHAR(x)	VARCHAR(y)	VARCHAR(z) donde z = max(x,y)
VARCHAR(x)	CHAR(y) o VARCHAR(y)	VARCHAR(z) donde z = max(x,y)
LONG VARCHAR	CHAR(y), VARCHAR(y) o LONG VARCHAR	LONG VARCHAR
CLOB(x)	CHAR(y), VARCHAR(y) o CLOB(y)	CLOB(z) donde z = max(x,y)
CLOB(x)	LONG VARCHAR	CLOB(z) donde z = max(x,32700)

La página de códigos de la serie de caracteres del resultado se derivará en base a las reglas de conversión de series.

Series gráficas

Las series gráficas son compatibles con otras series gráficas. Las series gráficas incluyen los tipos de datos GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC y DBCLOB.

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
GRAPHIC(x)	GRAPHIC(y)	GRAPHIC(z) donde z = max(x,y)
VARGRAPHIC(x)	GRAPHIC(y) o VARGRAPHIC(y)	VARGRAPHIC(z) donde z = max(x,y)
LONG VARGRAPHIC	GRAPHIC(y), VARGRAPHIC(y) o LONG VARGRAPHIC	LONG VARGRAPHIC
DBCLOB(x)	GRAPHIC(y), VARGRAPHIC(y) o DBCLOB(y)	DBCLOB(z) donde z = max (x,y)
DBCLOB(x)	LONG VARGRAPHIC	DBCLOB(z) donde z = max (x,16350)

La página de códigos de la serie gráfica del resultado se derivará en base a las reglas de conversión de series.

Series de caracteres y gráficas en una base de datos Unicode

En una base de datos Unicode, las series de caracteres y las series gráficas son compatibles.

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
GRAPHIC(x)	CHAR(y) o GRAPHIC(y)	GRAPHIC(z) donde z = max(x,y)
VARGRAPHIC(x)	CHAR(y) o VARCHAR(y)	VARGRAPHIC(z) donde z = max(x,y)
VARCHAR(x)	GRAPHIC(y) o VARGRAPHIC	VARGRAPHIC(z) donde z = max(x,y)
LONG VARGRAPHIC	CHAR(y) o VARCHAR(y) o LONG VARCHAR	LONG VARGRAPHIC
LONG VARCHAR	GRAPHIC(y) o VARGRAPHIC(y)	LONG VARGRAPHIC
DBCLOB(x)	CHAR(y) o VARCHAR(y) o CLOB(y)	DBCLOB(z) donde z = max (x,y)
DBCLOB(x)	LONG VARCHAR	DBCLOB(z) donde z = max (x,16350)
CLOB(x)	GRAPHIC(y) o VARGRAPHIC(y)	DBCLOB(z) donde z = max (x,y)
CLOB(x)	LONG VARGRAPHIC	DBCLOB(z) donde z = max (x,16350)

Objeto grande binario (BLOB)

Un BLOB sólo es compatible con otro BLOB y el resultado es un BLOB. La función escalar BLOB puede utilizarse para convertir desde otros tipos si éstos deben tratarse como tipos BLOB. La longitud del resultado BLOB es la longitud mayor de todos los tipos de datos.

Numérico

Los tipos numéricos son compatibles con otros tipos numéricos. Los tipos numéricos incluyen SMALLINT, INTEGER, BIGINT,DECIMAL, REAL y DOUBLE.

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
SMALLINT	SMALLINT	SMALLINT
INTEGER	INTEGER	INTEGER
INTEGER	SMALLINT	INTEGER
BIGINT	BIGINT	BIGINT
BIGINT	INTEGER	BIGINT
BIGINT	SMALLINT	BIGINT
DECIMAL(w,x)	SMALLINT	DECIMAL(p,x) donde p = x+max(w-x,5)[1]
DECIMAL(w,x)	INTEGER	DECIMAL(p,x) donde p = x+max(w-x,11)[1]
DECIMAL(w,x)	BIGINT	DECIMAL(p,x) donde p = x+max(w-x,19)[1]
DECIMAL(w,x)	DECIMAL(y,z)	DECIMAL(p,s) donde p = max(x,z)+max(w-x,y-z)[1]s = max(x,z)
REAL	REAL	REAL
REAL	DECIMAL, BIGINT, INTEGER o SMALLINT	DOUBLE
DOUBLE	cualquier tipo numérico	DOUBLE

[1] La precisión no puede exceder de 31.

DATE
Una fecha es compatible con otra fecha o con cualquier expresión CHAR o VARCHAR que contiene una representación de serie válida de una fecha. El tipo de datos del resultado es DATE.

TIME
Una hora es compatible con otra hora o con cualquier expresión CHAR o VARCHAR que contenta una representación de serie válida de una hora. El tipo de datos del resultado es TIME.

TIMESTAMP
Una indicación de fecha y hora es compatible con otra indicación de fecha y hora o con cualquier expresión CHAR o VARCHAR que contenga una representación de serie válida de una indicación de fecha y hora. El tipo de datos del resultado es TIMESTAMP.

DATALINK
Un enlace de datos es compatible con otro enlace de datos. El tipo de datos del resultado es DATALINK. La longitud del resultado DATALINK es la longitud mayor de todos los tipos de datos.

Tipos definidos por el usuario

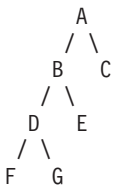
Tipos diferenciados: Un tipo diferenciado definido por el usuario sólo es compatible con el mismo tipo diferenciado definido por el usuario. El tipo de datos del resultado es el tipo diferenciado definido por el usuario.

Tipos de referencia: Un tipo de referencia es compatible con otro tipo de referencia en el caso de que sus tipos de destino tengan un supertipo común. El

tipo de datos del resultado es un tipo de referencia que tiene un supertipo común como tipo de destino. Si todos los operandos tienen la tabla de ámbito idéntica, el resultado tiene esta tabla de ámbito. De lo contrario, el resultado no tiene ámbito.

Tipos estructurados: Un tipo estructurado es compatible con otro tipo de estructurado siempre tengan un supertipo común. El tipo de datos estático de la columna del tipo estructurado resultante es el tipo estructurado que es el supertipo menos común de cualquiera de las dos columnas.

Por ejemplo, considere la siguiente jerarquía de tipos estructurados:



Los tipos estructurados del tipo estático E y F son compatibles con el tipo estático resultante de B, que es el supertipo menos común de E y F.

Atributo con posibilidad de nulos del resultado

A excepción de INTERSECT y EXCEPT, el resultado permite nulos a menos que ambos operandos no permitan nulos.

- Para INTERSECT, si cualquier operando no permite nulos, el resultado no permite nulos (la intersección nunca sería nula).
- Para EXCEPT, si el primer operando no permite nulos, el resultado no permite nulos (el resultado sólo puede ser valores del primer operando).

Información relacionada:

- "BLOB" en la página 302
- "Reglas para la conversión de series" en la página 129

Reglas para la conversión de series

La página de códigos utilizada para realizar una operación se determina por las reglas que se aplican a los operandos en dicha operación. Esta sección explica dichas reglas.

Estas reglas se aplican a:

- Las columnas de serie correspondientes en selecciones completas con operaciones de conjuntos (UNION, INTERSECT y EXCEPT)
- Los operandos de concatenación
- Los operandos de predicados (a excepción de LIKE)
- Expresiones resultantes de una expresión CASE
- Los argumentos de la función escalar COALESCE (y VALUE)
- Los valores de expresiones de la lista de entrada de un predicado IN
- Las expresiones correspondientes de una cláusula VALUES de múltiples filas.

En cada caso, la página de códigos del resultado se determina en el momento del enlace, y la ejecución de la operación puede implicar la conversión de series a la página de códigos identificada por dicha página de códigos. Un carácter que no tenga una conversión válida se correlaciona con el carácter de sustitución del juego de caracteres y SQLWARN10 se establece en 'W' en la SQLCA.

La página de códigos del resultado se determina por las páginas de códigos de los operandos. Las páginas de códigos de los dos primeros operandos determinan una página de códigos del resultado intermedia, esta página de códigos y la del siguiente operando determinan una nueva página de códigos del resultado intermedia (si se aplica), etcétera. La última página de códigos del resultado intermedia y la página de códigos del último operando determinan la página de códigos de la serie o columna del resultado. En cada par de páginas de códigos, el resultado se determina por la aplicación secuencial de las reglas siguientes:

- Si las páginas de códigos son iguales, el resultado es dicha página de códigos.
- Si cualquiera de las dos páginas de códigos es BIT DATA (página de códigos 0), la página de códigos del resultado es BIT DATA.
- En una base de datos Unicode, si una página de códigos denota datos en un esquema de codificación que es distinto de la otra página de códigos, el resultado es UCS-2 sobre UTF-8 (es decir, los datos de tipo gráfico sobre los datos de tipo carácter). (En una base de datos que no sea Unicode, no se permite la conversión entre distintos esquemas de codificación.
- Para los operandos que son variables del lenguaje principal (cuya página de códigos no es BIT DATA), la página de códigos resultante es la página de códigos de la base de datos. Los datos de entrada de este tipo de variables del lenguaje principal se convierten de la página de códigos de la aplicación a la página de códigos de la base de datos antes de utilizarse.

Las conversiones a la página de códigos del resultado se realizan, si es necesario, para:

- Un operando del operador de concatenación
- El argumento seleccionado de la función escalar COALESCE (o VALUE)
- La expresión del resultado seleccionada de la expresión CASE
- Las expresiones de la lista in del predicado IN
- Las expresiones correspondientes de una cláusula VALUES de múltiples filas

- Las columnas correspondientes que hacen referencia en operaciones de conjuntos.

La conversión de los caracteres es necesaria si son ciertas todas las afirmaciones siguientes:

- Las páginas de códigos son diferentes
- Ninguna serie es BIT DATA
- La serie no es nula ni está vacía

Ejemplos

Ejemplo 1: Supongamos lo siguiente en una base de datos creada con la página de códigos 850:

Expresión	Tipo	Página de códigos
COL_1	columna	850
HV_2	variable del lenguaje principal	437

Cuando se evalúa el predicado:

```
COL_1 CONCAT :HV_2
```

la página de códigos del resultado de los dos operandos es 850, porque los datos de variables del lenguaje principal se convertirán a página de códigos de la base de datos antes de utilizarse.

Ejemplo 2: Utilizando la información del ejemplo anterior para evaluar el predicado:

```
COALESCE(COL_1, :HV_2:NULLIND,)
```

la página de códigos del resultado es 850. Por lo tanto, la página de códigos del resultado para la función escalar COALESCE será la página de códigos 850.

Tipos de datos compatibles entre particiones

La *compatibilidad entre particiones* se define entre los tipos de base de datos de las columnas correspondientes de las claves de particionamiento. Los tipos de datos compatibles entre particiones tienen la propiedad de que dos variables, una de cada tipo, con el mismo valor, se correlacionan con el mismo índice de mapa de particionamiento por la misma función de partición.

La Tabla 10 muestra la compatibilidad de los tipos de datos en particiones.

La compatibilidad entre particiones tiene las características siguientes:

- Se utilizan formatos internos para DATE, TIME y TIMESTAMP. No son compatibles entre sí y ninguno es compatible con CHAR.
- La compatibilidad entre particiones no se ve afectada por las columnas con definiciones NOT NULL o FOR BIT DATA.
- Los valores NULL de los tipos de datos compatibles se tratan de manera idéntica. Se pueden generar resultados diferentes para los valores NULL de tipos de datos no compatibles.
- Se utiliza el tipo de datos base del UDT para analizar la compatibilidad entre particiones.
- Los decimales del mismo valor de la clave de particionamiento se tratan de manera idéntica, incluso si difieren su escala y precisión.
- La función de generación aleatoria proporcionada por el sistema ignora los blancos de cola de las series de caracteres (CHAR, VARCHAR, GRAPHIC o VARGRAPHIC).
- CHAR o VARCHAR de diferentes longitudes son tipos de datos compatibles.
- Los valores REAL o DOUBLE se tratan de manera idéntica incluso si su precisión es diferente.

Tabla 10. Compatibilidades entre particiones

Operandos	Entero binario	Número decimal	Coma flotante	Serie de caracteres	Serie gráfica	Fecha	Hora	Indicación de fecha y hora	Tipo diferenciado	Tipo estructurado
Entero binario	Sí	No	No	No	No	No	No	No	[1]	No
Número decimal	No	Sí	No	No	No	No	No	No	[1]	No
Coma flotante	No	No	Sí	No	No	No	No	No	[1]	No
Serie de caracteres[3]	No	No	No	Sí[2]	No	No	No	No	[1]	No
Serie gráfica[3]	No	No	No	No	Sí	No	No	No	[1]	No
Fecha	No	No	No	No	No	Sí	No	No	[1]	No
Hora	No	No	No	No	No	No	Sí	No	[1]	No
Indicación de fecha y hora	No	No	No	No	No	No	No	Sí	[1]	No
Tipo diferenciado	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	No
Tipo estructurado[3]	No	No	No	No	No	No	No	No	No	No

Tabla 10. Compatibilidades entre particiones (continuación)

Operandos	Entero binario	Número decimal	Coma flotante	Serie de caracteres	Serie gráfica	Fecha	Hora	Indicación de fecha y hora	Tipo diferenciado	Tipo estructurado

Nota:

1. El valor de un tipo diferenciado definido por el usuario (UDT) es compatible, a nivel de partición, con el tipo fuente del UDT o cualquier otro UDT con un tipo fuente compatible a nivel de partición.
2. El atributo FOR BIT DATA no afecta a la compatibilidad entre particiones.
3. Observe que los tipos estructurados definidos por el usuario y los tipos de datos LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB y BLOB no son aplicables para la compatibilidad de particiones, pues no están soportados en las claves de particionamiento.

Constantes

Una *constante* (a veces llamada un *literal*) especifica un valor. Las constantes se clasifican en constantes de tipo serie y constantes numéricas. Las constantes numéricas pueden, a su vez, ser constantes enteras, de coma flotante y decimales.

Todas las constantes tienen el atributo NOT NULL.

Un valor de cero negativo en una constante numérica (-0) indica el mismo valor que un cero sin el signo (0).

Los tipos definidos por el usuario son difíciles de escribir. Esto significa que un tipo definido por el usuario sólo es compatible con su propio tipo. Sin embargo, una constante tiene un tipo interno. Por lo tanto, una operación que implique un tipo definido por el usuario y una constante sólo es posible si el tipo definido por el usuario se ha convertido al tipo interno de la constante o si la constante se ha convertido al tipo definido por el usuario. Por ejemplo, si se utiliza la tabla y el tipo diferenciado del apartado “Comparaciones de tipos definidos por el usuario” en la página 123, serán válidas las siguientes comparaciones con la constante 14:

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > CAST(14 AS YOUTH)

SELECT * FROM CAMP_DB2_ROSTER
  WHERE CAST(AGE AS INTEGER) > 14
```

No será válida la siguiente comparación:

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > 14
```

Constantes enteras

Una *constante entera* especifica un entero en forma de número, con signo o sin signo, con un máximo de 19 dígitos que no incluye ninguna coma decimal. El tipo de datos de una constante entera es entero grande si su valor está comprendido en el rango de un entero grande. El tipo de datos de una constante entera es un entero superior si su valor se encuentra fuera del rango de un entero grande, pero está comprendido en el rango de un entero superior. Una constante definida fuera del rango de valores enteros superior se considera una constante decimal.

Observe que la representación literal más pequeña de una constante entera grabde es -2 147 483 647, y no -2 147 483 648, que es el límite para los valores enteros. De manera similar, la representación literal más pequeña de una constante entera superior es -9 223 372 036 854 775 807, y no -9 223 372 036 854 775 808, que es el límite para los valores enteros superiores.

Ejemplos:

```
64     -15     +100     32767     720176      12345678901
```

En los diagramas de sintaxis, el término 'entero' se utiliza para una constante entera grande que no debe incluir un signo.

Constantes de coma flotante

Una *constante de coma flotante* especifica un número de coma flotante en forma de dos números separados por una E. El primer número puede incluir un signo y una coma decimal; el segundo número puede incluir un signo, pero no una coma decimal. El tipo de datos de una constante de coma flotante es de precisión doble.

El valor de la constante es el producto del primer número y la potencia de 10 especificada por el segundo número; este valor debe estar dentro del rango de los números de coma flotante. El número de caracteres de la constante no debe exceder de 30.

Ejemplos:

```
15E1     2,E5     2,2E-1     +5,E+2
```

Constantes decimales

Una *constante decimal* es un número con o sin signo de 31 dígitos de longitud como máximo y que incluye una coma decimal o no está comprendido dentro del rango de enteros binarios. Debe estar comprendido en el rango de números decimales. La precisión es el número total de dígitos (incluyendo los ceros iniciales y de cola); la escala es el número de dígitos situados a la derecha de la coma decimal (incluyendo los ceros de cola).

Ejemplos:

```
25,5     1000,     -15,     +37589,3333333333
```

Constantes de series de caracteres

Una *constante de serie de caracteres* especifica una serie de caracteres de longitud variable y consta de una secuencia de caracteres que empieza y finaliza por un apóstrofo ('). Esta modalidad de constante de tipo serie especifica la serie de caracteres contenida entre los delimitadores de series. La longitud de la serie de caracteres no debe ser mayor que 32.672 bytes. Se utilizan dos delimitadores de series consecutivos para representar un delimitador de series dentro de la serie de caracteres.

Ejemplos:

```
'12/14/1985'
'32'
'DON''T CHANGE'
```

El valor de una constante se convierte siempre en la página de códigos de la base de datos cuando se vincula con la base de datos. Se considera que está en la página de códigos de la base de datos. Por lo tanto, si se utiliza en una expresión que combina una constante con una columna FOR BIT DATA y cuyo resultado es FOR BIT DATA, el valor de la constante *no* se convertirá desde su representación de página de códigos de base de datos.

Constantes hexadecimales

Una *constante hexadecimal* especifica una serie de caracteres de longitud variable de la página de códigos de la sección.

El formato de una constante hexadecimal es una X seguida por una secuencia de caracteres que empieza y termina con un apóstrofo ('). Los caracteres que se incluyen entre los apóstrofos deben corresponder a una cantidad par de dígitos hexadecimales. El número de dígitos hexadecimales no debe exceder de 16.336 o, de lo contrario, se producirá un error (SQLSTATE -54002). Un dígito hexadecimal representa 4 bits. Se especifica como un dígito o cualquiera de las letras de la ″A″ a la ″F″ (en mayúsculas o en minúsculas), donde A representa el patrón de bits '1010', B representa '1011', etc. Si una constante hexadecimal no tiene el formato correcto (por ejemplo, contiene un dígito hexadecimal no válido o un número impar de dígitos hexadecimales), se genera un error (SQLSTATE 42606).

Ejemplos:

```
X'FFFF'       que representa el patrón de bits    '1111111111111111'

X'4672616E6B' que representa el patrón VARCHAR de la serie ASCII 'Frank'
```

Constantes de series gráficas

Una *constante de serie gráfica* especifica una serie gráfica de longitud variable que consta de una secuencia de caracteres de doble byte que empieza y termina con un apóstrofo de un solo byte (') y que está precedida por una G o una N de un solo byte. Los caracteres entre los apóstrofos deben representar un número par de bytes y la longitud de la serie gráfico no debe sobrepasar los 16 336 bytes.

Ejemplos:

```
G'serie de caracteres de doble byte'
N'serie de caracteres de doble byte'
```

El apóstrofo no debe aparecer como parte de un carácter MBCS para que se considere como delimitador.

Información relacionada:

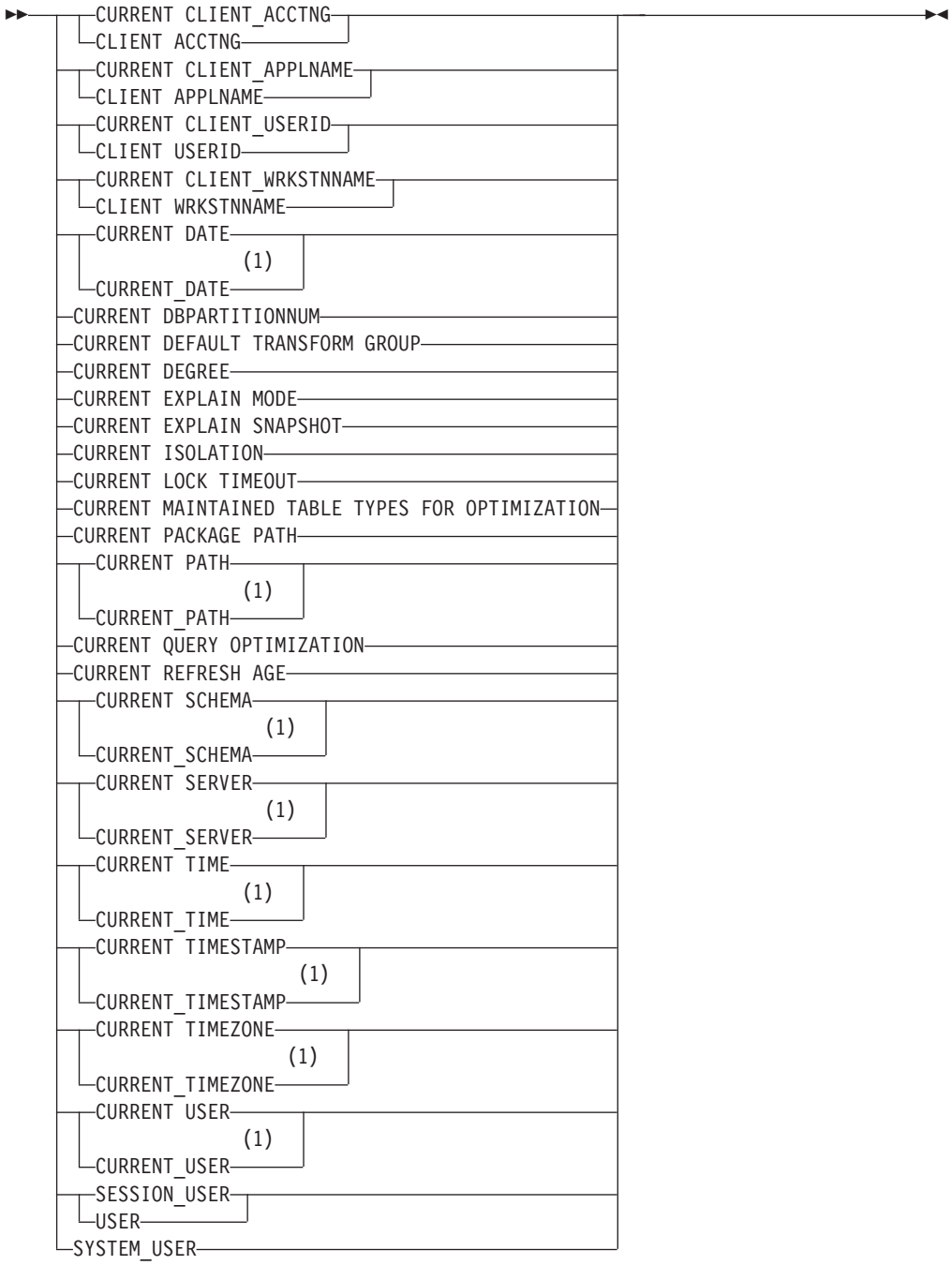
- "Expresiones" en la página 181
- "Asignaciones y comparaciones" en la página 110

Registros especiales

Registros especiales

Registros especiales

Un *registro especial* es un área de almacenamiento que el gestor de bases de datos define para un proceso de aplicación. Se utiliza para almacenar información a la que se puede hacer referencia en sentencias de SQL. Una referencia a un registro especial es una referencia a un valor proporcionado por el servidor actual. Si el valor es una serie, su CCSID es el CCSID por omisión del servidor actual. Se hace referencia a los registros especiales de la forma siguiente:



Notas:

1 El estándar básico de 1999 de SQL utiliza el formato con el subrayado.

Algunos registros especiales puede actualizarse utilizando la sentencia SET. La tabla siguiente muestra cuáles de estos registros especiales pueden actualizarse.

Tabla 11. Registros especiales

Registro especial	Actualizable
CURRENT CLIENT_ACCTNG	No
CURRENT CLIENT_APPLNAME	No
CURRENT CLIENT_USERID	No
CURRENT CLIENT_WRKSTNNAME	No
CURRENT DATE	No
CURRENT DBPARTITIONNUM	No
CURRENT DEFAULT TRANSFORM GROUP	Sí
CURRENT DEGREE	Sí
CURRENT EXPLAIN MODE	Sí
CURRENT EXPLAIN SNAPSHOT	Sí
CURRENT ISOLATION	Sí
CURRENT LOCK TIMEOUT	Sí
CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION	Sí
CURRENT PACKAGE PATH	Sí
CURRENT PATH	Sí
CURRENT QUERY OPTIMIZATION	Sí
CURRENT REFRESH AGE	Sí
CURRENT SCHEMA	Sí
CURRENT SERVER	No
CURRENT TIME	No
CURRENT TIMESTAMP	No
CURRENT TIMEZONE	No
CURRENT USER	No
SESSION_USER	Sí
SYSTEM_USER	No
USER	Sí

Cuando se hace referencia a un registro especial en una rutina, el valor del registro especial de la rutina depende de si el registro especial es actualizable o no. Para registros especiales no actualizables, el valor se define en el valor por omisión del registro especial. Para registros especiales actualizables, el valor inicial se hereda del invocador de la rutina y puede modificarse con una sentencia SET posterior dentro de la rutina.

Registro especial CURRENT CLIENT_ACCTNG

El registro especial CURRENT CLIENT_ACCTNG (o CLIENT ACCTNG) contiene el valor de la serie de contabilidad a partir de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor de la serie de contabilidad puede cambiarse utilizando la API para Establecer información de cliente (**sqleseti**).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo: Obtenga el valor actual de la serie de contabilidad para esta conexión.

```
VALUES (CURRENT CLIENT_ACCTNG)
   INTO :ACCT_STRING
```

Registro especial CURRENT CLIENT_APPLNAME

El registro especial CLIENT_APPLNAME (o CLIENT APPLNAME) contiene el valor del nombre de la aplicación a partir de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor del nombre de aplicación puede cambiarse utilizando la API para Establecer información de cliente (**sqleseti**).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo: Seleccione los departamentos a los que se les permite utilizar la aplicación que se está usando en esta conexión.

```
SELECT DEPT
  FROM DEPT_APPL_MAP
  WHERE APPL_NAME = CURRENT CLIENT_APPLNAME
```

Registro especial CURRENT CLIENT_USERID

El registro especial CLIENT_USERID (o CLIENT USERID) contiene el valor del ID de usuario de cliente a partir de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor del ID de usuario de cliente puede cambiarse utilizando la API para Establecer información de cliente (**sqleseti**).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo: Averigüe en qué departamento funciona el ID de usuario de cliente actual.

```
SELECT DEPT
  FROM DEPT_USERID_MAP
  WHERE USER_ID = CURRENT CLIENT_USERID
```

Registro especial CURRENT CLIENT_WRKSTNNAME

El registro especial CLIENT_WRKSTNNAME (o CLIENT WRKSTNNAME) contiene el valor del nombre de la estación de trabajo a partir de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor del nombre de estación de trabajo puede cambiarse utilizando la API para Establecer información de cliente (**sqleseti**).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo: Obtenga el nombre de estación de trabajo que se está utilizando para esta conexión.

```
VALUES (CURRENT CLIENT_WRKSTNNAME)
   INTO :WS_NAME
```

CURRENT DATE

El registro especial CURRENT DATE (o CURRENT_DATE) especifica una fecha basada en la lectura del reloj cuando se ejecuta la sentencia de SQL en el servidor de aplicaciones. Si este registro especial se utiliza más de una vez en la misma sentencia de SQL o bien con CURRENT TIME o CURRENT TIMESTAMP en una sola sentencia, todos los valores se basan en la misma lectura del reloj.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT DATE no se hereda de la sentencia que la invoca.

En un sistema federado, CURRENT DATE se puede utilizar en una consulta prevista para fuentes de datos. Cuando se procesa la consulta, la fecha devuelta se obtendrá del registro CURRENT DATE, en el servidor federado, no de las fuentes de datos.

Ejemplo: Utilizando la tabla PROJECT, establezca la fecha final del proyecto (PRENDATE) del proyecto MA2111 (PROJNO) en la fecha actual.

```
UPDATE PROJECT
  SET PRENDATE = CURRENT DATE
  WHERE PROJNO = 'MA2111'
```

CURRENT DBPARTITIONNUM

El registro especial CURRENT DBPARTITIONNUM especifica un valor INTEGER que identifica el número de nodo coordinador de la sentencia. Para las sentencias emitidas desde una aplicación, el coordinador es la partición a la que se conecta la aplicación. Para las sentencias emitidas desde una rutina, el coordinador es la partición a la que se invoca la rutina.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT DBPARTITIONNUM nunca se hereda de la sentencia que la invoca.

CURRENT DBPARTITIONNUM devuelve 0 si la instancia de base de datos no está definida para soportar el particionamiento. (En otras palabras, si no hay ningún archivo db2nodes.cfg. Para las bases de datos particionadas, el archivodb2nodes.cfg existe y contiene las definiciones de las particiones o los nodos).

Es posible cambiar CURRENT DBPARTITIONNUM mediante la sentencia CONNECT, pero sólo bajo ciertas condiciones.

Para compatibilidad con versiones anteriores a la Versión 8, la palabra clave NODE puede sustituirse por DBPARTITIONNUM.

Ejemplo: Establezca la variable del lenguaje principal APPL_NODE (entero) en el número de la partición a la que está conectada la aplicación.

```
VALUES CURRENT DBPARTITIONNUM
  INTO :APPL_NODE
```

Información relacionada:

- "Sentencia CONNECT (Tipo 1)" en la publicación *Consulta de SQL, Volumen 2*

CURRENT DEFAULT TRANSFORM GROUP

El registro especial CURRENT DEFAULT TRANSFORM GROUP especifica un valor VARCHAR(18) que identifica el nombre del grupo de transformación utilizado por las sentencias de SQL dinámicas para intercambiar valores de tipo estructurado definidos por el usuario con programas de lenguaje principal. Este registro especial no especifica los grupos de transformación utilizados en las sentencias de SQL dinámico o en el intercambio de parámetros y resultados con funciones externas o métodos.

Su valor puede definirse mediante la sentencia SET CURRENT DEFAULT TRANSFORM GROUP. Si no se define ningún valor, el valor inicial del registro especial es la serie vacía (VARCHAR con una longitud de cero).

En un sentencia de SQL dinámico (es decir, una sentencia que interacciona con variables del lenguaje principal), el nombre del grupo de transformación utilizado para intercambiar valores es el mismo que el nombre de este registro especial, a menos que el registro contenga la serie vacía. Si el registro contiene la serie vacía (no se ha definido ningún valor utilizando la sentencia SET CURRENT DEFAULT TRANSFORM GROUP), se utiliza el grupo de transformación DB2_PROGRAM para la transformación. Si el grupo de transformación DB2_PROGRAM no está definido para el tipo estructurado indicado, se emite un error durante la ejecución (SQLSTATE 42741).

Ejemplos:

Establezca el grupo de transformación por omisión en MYSTRUCT1. Las funciones TO SQL y FROM SQL definidas en la transformación MYSTRUCT1 se utilizan para intercambiar variables de tipo estructurado, definidas por el usuario, con el programa de lenguaje principal.

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```

Recupere el nombre del grupo de transformación por omisión asignado a este registro especial.

```
VALUES (CURRENT DEFAULT TRANSFORM GROUP)
```

CURRENT DEGREE

El registro especial CURRENT DEGREE especifica el grado de paralelismo intrapartición para la ejecución de sentencias de SQL dinámico. (Para SQL estático, la opción de vinculación de DEGREE proporciona el mismo control.) El tipo de datos del registro es CHAR(5). Los valores válidos son ANY o la representación de serie de un entero entre 1 y 32 767, inclusive.

Si el valor de CURRENT DEGREE representado como un entero es 1 cuando una sentencia de SQL se prepara dinámicamente, la ejecución de esta sentencia no utilizará el paralelismo intrapartición.

Si el valor de CURRENT DEGREE representado como un entero es mayor que 1 y menor o igual a 32 767 cuando una sentencia de SQL se prepara dinámicamente, la ejecución de esta sentencia puede implicar el paralelismo intrapartición con el grado especificado.

Si el valor de CURRENT DEGREE es ANY cuando una sentencia de SQL se prepara dinámicamente, la ejecución de dicha sentencia puede implicar el paralelismo intrapartición que utiliza un grado determinado por el gestor de bases de datos.

El grado de paralelismo real durante la ejecución será el menor de los valores siguientes:

- El valor del parámetro de configuración del grado de consulta máximo(*max_querydegree*)
- El grado de ejecución de la aplicación
- El grado de compilación de la sentencia de SQL.

Si el parámetro de configuración del gestor de bases de datos *intra_parallel* se establece en NO, el valor del registro especial CURRENT DEGREE se ignorará con el fin de optimizar y la sentencia no utilizará el paralelismo intrapartición.

El valor puede cambiarse invocando la sentencia SET CURRENT DEGREE.

El valor inicial de CURRENT DEGREE lo determina el parámetro de configuración de la base de datos *dft_degree*.

Información relacionada:

- "Sentencia SET CURRENT DEGREE" en la publicación *Consulta de SQL, Volumen 2*

CURRENT EXPLAIN MODE

El registro especial CURRENT EXPLAIN MODE contiene un valor VARCHAR(254) que controla la actuación del recurso Explain con respecto a sentencias de SQL dinámico admisibles. Este recurso genera e inserta información de Explain en las tablas de Explain. Esta información no incluye la instantánea de Explain. Los valores posibles son YES, EXPLAIN, NO, REOPT, RECOMMEND INDEXES y EVALUATE INDEXES.(Para SQL estático, la opción de vinculación de EXPLAIN proporciona el mismo control. En el caso de los mandatos PREP y BIND, los valores de la opción EXPLAIN son: YES, NO y ALL.)

YES Habilita el recurso Explain y hace que la información de Explain para una sentencia de SQL dinámico se capture al compilar la sentencia.

EXPLAIN
Se habilita el recurso pero no se ejecutan las sentencias dinámicas.

NO Inhabilita el recurso Explain.

REOPT
Habilita el recurso Explain y hace que la información de Explain para una sentencia de SQL dinámico (o de vinculación incremental) sólo se capture cuando se reoptimice la sentencia utilizando valores reales para las variables de entrada (variables del lenguaje principal, registros especiales o marcadores de parámetros).

RECOMMEND INDEXES
Recomienda un conjunto de índices para cada consulta dinámica. Llena la tabla ADVISE_INDEX con el conjunto de índices.

EVALUATE INDEXES
Explica las consultas dinámicas como si existieran los índices recomendados. Los índices se seleccionan de la tabla ADVISE_INDEX.

El valor inicial es NO. Este valor puede cambiarse invocando la sentencia SET CURRENT EXPLAIN MODE.

Los valores de los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT interactúan al invocar el recurso Explain. El registro especial CURRENT EXPLAIN MODE también interactúa con la opción de vinculación EXPLAIN. RECOMMEND INDEXES y EVALUATE INDEXES sólo se pueden establecer para el registro CURRENT EXPLAIN MODE y deben establecerse utilizando la sentencia SET CURRENT EXPLAIN MODE.

Ejemplo: Establezca la variable del lenguaje principal EXPL_MODE (VARCHAR(254)) en el valor que hay actualmente en el registro especial CURRENT EXPLAIN MODE.

```
VALUES CURRENT EXPLAIN MODE
     INTO :EXPL_MODE
```

Información relacionada:

- "Sentencia SET CURRENT EXPLAIN MODE" en la publicación *Consulta de SQL, Volumen 2*
- Apéndice J, "Valores de los registros de EXPLAIN", en la página 777

CURRENT EXPLAIN SNAPSHOT

El registro especial CURRENT EXPLAIN SNAPSHOT contiene un valor CHAR(8) que controla el comportamiento del recurso de instantáneas de Explain. Este recurso genera información comprimida que incluye información sobre planes de acceso, costes del operador y estadísticas en tiempo de vinculación.

Sólo las siguientes sentencias tienen en cuenta el valor de este registro: DELETE, INSERT, SELECT, SELECT INTO, UPDATE, VALUES y VALUES INTO. Los valores posibles son YES, EXPLAIN, NO y REOPT. (Para SQL estático, la opción de vinculación EXPLSNAP proporciona el mismo control. En el caso de los mandatos PREP y BIND, los valores de la opción EXPLSNAP son: YES, NO y ALL.)

YES Habilita el recurso de instantáneas de Explain y realiza una instantánea de la representación interna de una sentencia de SQL dinámico al compilar la sentencia.

EXPLAIN
Habilita el recurso de instantáneas de Explain pero no se ejecutan las sentencias dinámicas.

NO Inhabilita el recurso de instantáneas de Explain.

REOPT
Habilita el recurso Explain y hace que la información de Explain para una sentencia de SQL dinámico (o de vinculación incremental) sólo se capture cuando se reoptimice la sentencia utilizando valores reales para las variables de entrada (variables del lenguaje principal, registros especiales o marcadores de parámetros).

El valor inicial es NO. Este valor puede cambiarse invocando la sentencia SET CURRENT EXPLAIN SNAPSHOT.

Los valores de los registros especiales CURRENT EXPLAIN SNAPSHOT y CURRENT EXPLAIN MODE interactúan al invocar el recurso Explain. El registro especial CURRENT EXPLAIN SNAPSHOT también interactúa con la opción de vinculación EXPLSNAP.

Ejemplo: Establezca la variable del lenguaje principal EXPL_SNAP (char(8)) en el valor que contiene actualmente el registro especial CURRENT EXPLAIN SNAPSHOT.

```
VALUES CURRENT EXPLAIN SNAPSHOT
  INTO :EXPL_SNAP
```

Información relacionada:

- "Sentencia SET CURRENT EXPLAIN SNAPSHOT" en la publicación *Consulta de SQL, Volumen 2*
- Apéndice J, "Valores de los registros de EXPLAIN", en la página 777

CURRENT ISOLATION

El registro especial CURRENT ISOLATION mantiene un valor CHAR(2) que identifica el nivel de aislamiento (en relación a otras sesiones simultáneas) correspondiente a las sentencias de SQL dinámico emitidas desde la sesión actual.

Los valores posibles son:

(blancos)
Sin definir; se utiliza el atributo de aislamiento del paquete.

UR Lectura no confirmada

CS Estabilidad del cursor

RR Lectura repetible

RS Estabilidad de lectura

El valor del registro especial CURRENT ISOLATION se puede cambiar mediante la sentencia SET CURRENT ISOLATION.

Hasta que se emita una sentencia SET CURRENT ISOLATION en una sesión o después de especificar RESET para SET CURRENT ISOLATION, el registro especial CURRENT ISOLATION está establecido en blancos y no se aplica a sentencias de SQL dinámico; el nivel de aislamiento utilizado se toma del atributo de aislamiento del paquete que ha emitido la sentencia de SQL dinámico. Una vez emitida una sentencia SET CURRENT ISOLATION, el registro especial CURRENT ISOLATION proporciona el nivel de aislamiento correspondiente a cualquier sentencia de SQL dinámico siguiente compilada dentro de la sesión, con independencia de los valores del paquete que emita la sentencia. Esto permanecerá vigente hasta que finalice la sesión o hasta que se emita una sentencia SET CURRENT ISOLATION con la opción RESET.

Ejemplo: Establecer para la variable del lenguaje principal ISOLATION_MODE (CHAR(2)) el valor actualmente almacenado en el registro especial CURRENT ISOLATION.

```
VALUES CURRENT ISOLATION
  INTO :ISOLATION_MODE
```

Conceptos relacionados:

- "Niveles de aislamiento" en la página 15

Información relacionada:

- "Sentencia SET CURRENT ISOLATION" en la publicación *Consulta de SQL, Volumen 2*

Registro especial CURRENT LOCK TIMEOUT

El registro especial CURRENT LOCK TIMEOUT especifica el número de segundos que debe esperarse un bloqueo antes de devolver un error que indique que no es posible obtener un bloqueo. Este registro especial afecta la fila, la tabla, la clave de índice y los bloqueos de bloques MDC. El tipo de datos del registro es INTEGER.

Los valores válidos para el registro especial CURRENT LOCK TIMEOUT son los enteros comprendidos entre -1 y 32767, ambos inclusive. Este registro especial también puede establecerse en un valor nulo. Un valor de -1 especifica que no deben producirse tiempos de espera excedidos y que la aplicación debe esperar hasta que se libere el bloqueo o se detecte un punto muerto. Un valor de 0 especifica que la aplicación no debe esperar un bloqueo; si no es posible obtener un bloqueo, debe devolverse un error inmediatamente.

El valor del registro especial CURRENT LOCK TIMEOUT puede modificarse invocando la sentencia SET CURRENT LOCK TIMEOUT. Su valor inicial es nulo; en este caso, se utiliza el valor actual del parámetro de configuración de la base de datos *locktimeout* al esperar un bloqueo y se devolverá este valor para el registro especial.

Información relacionada:

- "Sentencia SET CURRENT LOCK TIMEOUT" en la publicación *Consulta de SQL, Volumen 2*

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

El registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION especifica un valor VARCHAR(254) que identifica los tipos de tablas que pueden tenerse en cuenta al optimizar el proceso de las series de SQL dinámico. Las consultas de SQL incorporado estático nunca tienen en cuenta las tablas de consulta materializadas.

El valor inicial de CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION es SYSTEM. Su valor puede cambiarse con la sentencia SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION.

Información relacionada:

- "Sentencia SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION" en la publicación *Consulta de SQL, Volumen 2*
- "dft_mttb_types - Default maintained table types for optimization configuration parameter" en la publicación *Administration Guide: Performance*

Registro especial CURRENT PACKAGE PATH

El registro especial CURRENT PATH especifica un valor VARCHAR(4096) que identifica la vía de acceso que se ha de utilizar para resolver las referencias a paquetes que son necesarias al ejecutar sentencias de SQL.

El valor puede ser una serie vacía o en blanco o una lista de uno o varios nombres de esquema delimitados por comillas dobles y separados por comas. Las comillas dobles que aparezcan como parte de la serie deberán representarse como dos comillas dobles, como suele hacerse con los identificadores delimitados. Los delimitadores y las comas contribuyen a la longitud del registro especial.

Este registro especial se aplica a sentencias tanto estáticas como dinámicas.

El valor inicial de CURRENT PACKAGE PATH en una función, un método o un procedimiento definido por el usuario se hereda de la aplicación que lo invoca. En otros contextos, el valor inicial de CURRENT PACKAGE PATH es una serie vacía. El valor sólo es una lista de esquemas si el proceso de la aplicación ha especificado de forma explícita una lista de esquemas mediante la sentencia SET CURRENT PACKAGE PATH.

Ejemplos:

Una aplicación utilizará varios paquetes SQLJ(en los esquemas SQLJ1 y SQLJ2) y paquete JDBC (en el esquema DB2JAVA). El registro especial CURRENT PACKAGE PATH debe establecerse para comprobar SQLJ1, SQLJ2 y DB2JAVA, en este orden.

```
SET CURRENT PACKAGE PATH = "SQLJ1", "SQLJ2", "DB2JAVA"
```

La variable del lenguaje principal HVPKLIST debe establecerse en el valor almacenado actualmente en el registro especial CURRENT PACKAGE PATH.

```
VALUES CURRENT PACKAGE PATH INTO :HVPKLIST
```

Información relacionada:

- "CURRENT PATH" en la página 152
- "Sentencia SET CURRENT PACKAGE PATH" en la publicación *Consulta de SQL, Volumen 2*

CURRENT PATH

El registro especial CURRENT PATH (o CURRENT_PATH) especifica un valor VARCHAR(254) que identifica la vía de acceso de SQL que se ha de utilizar para resolver las referencias de funciones y las referencias de tipos de datos de sentencias de SQL preparadas dinámicamente. CURRENT FUNCTION PATH es sinónimo de CURRENT PATH. CURRENT PATH también se utiliza para resolver las referencias de procedimientos almacenados en sentencias CALL. El valor inicial es el valor por omisión que se especifica más abajo. Para SQL estático, la opción de vinculación FUNCPATH proporciona una vía de acceso de SQL que se utiliza para la resolución de funciones y tipos de datos.

El registro especial CURRENT PATH contiene una lista de uno o varios nombres de esquema escritos entre comillas dobles y separados por comas. Por ejemplo, una vía de acceso de SQL que especifica que el gestor de bases de datos primero debe mirar en el esquema FERMAT, luego en el esquema XGRAPHIC y por último en el esquema SYSIBM se devuelve en el registro especial CURRENT PATH de la siguiente manera:

```
"FERMAT","XGRAPHIC","SYSIBM"
```

El valor por omisión es“SYSIBM”,“SYSFUN”,“SYSPROC”,*X*, donde *X* es el valor del registro especial USER, delimitado por comillas dobles. El valor puede cambiarse invocando la sentencia SET CURRENT PATH. No es necesario especificar el esquema SYSIBM. Si no se incluye en la vía de acceso de SQL, se supone implícitamente que es el primer esquema. SYSIBM no toma ninguno de los 254 caracteres si se asume implícitamente.

Un tipo de datos que no está calificado con un nombre de esquema se calificará implícitamente con el primer esquema de la vía de acceso de SQL que contenga un tipo de datos con el mismo nombre no calificado. Existen excepciones a esta regla, como se indica en las descripciones de las sentencias siguientes: CREATE DISTINCT TYPE, CREATE FUNCTION, COMMENT y DROP.

Ejemplo: Utilizando la vista de catálogo SYSCAT.VIEWS, busque todas las vistas que se hayan creado con el mismo valor que el valor actual del registro especial CURRENT PATH.

```
SELECT VIEWNAME, VIEWSCHEMA FROM SYSCAT.VIEWS
  WHERE FUNC_PATH = CURRENT PATH
```

Información relacionada:

- “Funciones” en la página 164
- “Sentencia SET PATH” en la publicación *Consulta de SQL, Volumen 2*

CURRENT QUERY OPTIMIZATION

El registro especial CURRENT QUERY OPTIMIZATION especifica un valor INTEGER que controla la clase de optimización de consulta que realiza el gestor de bases de datos al vincular sentencias de SQL dinámico. La opción de vinculación QUERYOPT controla la clase de optimización de consulta para las sentencias de SQL estático. Los valores posibles oscilan entre 0 y 9. Por ejemplo, si la clase de optimización de consulta se establece en 0 (optimización mínima), el valor del registro especial es 0. El valor por omisión está determinado por el parámetro de configuración de la base de datos *dft_queryopt*. El valor puede cambiarse invocando la sentencia SET CURRENT QUERY OPTIMIZATION.

Ejemplo: Utilizando la vista de catálogo SYSCAT.PACKAGES, busque todos los planes que se han vinculado con el mismo valor que el valor actual del registro especial CURRENT QUERY OPTIMIZATION.

```
SELECT PKGNAME, PKGSCHEMA FROM SYSCAT.PACKAGES
   WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

Información relacionada:

- "Sentencia SET CURRENT QUERY OPTIMIZATION" en la publicación *Consulta de SQL, Volumen 2*

CURRENT REFRESH AGE

El registro especial CURRENT REFRESH AGE especifica un valor de duración de indicación de fecha y hora con un tipo de datos de DECIMAL(20,6). Es la duración máxima desde que se produjo un suceso de indicación de fecha y hora concreto en un objeto de datos de la antememoria (por ejemplo, una sentencia REFRESH TABLE procesada en una tabla de consulta materializada REFRESH DEFERRED mantenida por el sistema) de forma que el objeto de datos de la antememoria pueda utilizarse para optimizar el proceso de la consulta. Si CURRENT REFRESH AGE tiene un valor de 99 999 999 999 999 (ANY) y la clase de optimización de la consulta es 5 o superior, los tipos de tablas especificados en CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION se tienen en cuenta al optimizar el proceso de consulta de SQL dinámico.

El valor inicial de CURRENT REFRESH AGE es cero. El valor puede cambiarse invocando la sentencia SET CURRENT REFRESH AGE.

Información relacionada:

- "Sentencia SET CURRENT REFRESH AGE" en la publicación *Consulta de SQL, Volumen 2*

CURRENT SCHEMA

El registro especial CURRENT SCHEMA (o CURRENT_SCHEMA) especifica un valor VARCHAR(128) que identifica el nombre de esquema utilizado para calificar las referencias a objetos de base de datos, donde corresponda, en sentencias de SQL preparadas dinámicamente. Para conseguir la compatibilidad con DB2 para OS/390, CURRENT SQLID (o CURRENT_SQLID) es un sinónimo de CURRENT SCHEMA.

El valor inicial de CURRENT SCHEMA es el ID de autorización del usuario de la sesión actual. El valor puede cambiarse invocando la sentencia SET SCHEMA.

La opción de vinculación QUALIFIER controla el nombre de esquema utilizado para calificar las referencias a objetos de base de datos, donde corresponda, para sentencias de SQL estático.

Ejemplo: Establezca el esquema para la calificación de objetos en 'D123'.

```
SET CURRENT SCHEMA =  'D123'
```

CURRENT SERVER

El registro especial CURRENT SERVER (o CURRENT_SERVER) especifica un valor VARCHAR(18) que identifica el servidor de aplicaciones actual. El registro contiene el nombre real del servidor de aplicaciones, no un seudónimo.

Es posible cambiar CURRENT SERVER mediante la sentencia CONNECT, pero sólo bajo ciertas condiciones.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT SERVER no se hereda de la sentencia que la invoca.

Ejemplo: Establezca la variable del lenguaje principal APPL_SERVE (VARCHAR(18)) en el nombre del servidor de aplicaciones al que está conectada la aplicación.

```
VALUES CURRENT SERVER INTO :APPL_SERVE
```

Información relacionada:

- "Sentencia CONNECT (Tipo 1)" en la publicación *Consulta de SQL, Volumen 2*

CURRENT TIME

El registro especial CURRENT TIME (o CURRENT_TIME) especifica una hora basada en la lectura del reloj cuando se ejecuta la sentencia de SQL en el servidor de aplicaciones. Si este registro especial se utiliza más de una vez en la misma sentencia de SQL o bien con CURRENT DATE o CURRENT TIMESTAMP en una sola sentencia, todos los valores se basan en la misma lectura del reloj.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT TIME no se hereda de la sentencia que la invoca.

En un sistema federado, CURRENT TIME se puede utilizar en una consulta destinada a fuentes de datos. Cuando se procesa la consulta, la hora devuelta se obtendrá del registro CURRENT TIME del servidor federado, no de las fuentes de datos.

Ejemplo: Utilizando la tabla CL_SCHED, seleccione todas las clases (CLASS_CODE) que empiezan (STARTING) más tarde hoy. Las clases de hoy tienen un valor 3 en la columna DAY.

```
SELECT CLASS_CODE FROM CL_SCHED
  WHERE STARTING > CURRENT TIME AND DAY = 3
```

CURRENT TIMESTAMP

El registro especial CURRENT TIMESTAMP (o CURRENT_TIMESTAMP) especifica una indicación de fecha y hora basada en la lectura del reloj cuando se ejecuta la sentencia de SQL en el servidor de aplicaciones. Si este registro especial se utiliza más de una vez en la misma sentencia de SQL o bien con CURRENT DATE o CURRENT TIME en una sola sentencia, todos los valores se basan en la misma lectura del reloj.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT TIMESTAMP no se hereda de la sentencia que la invoca.

En un sistema federado, CURRENT TIMESTAMP se puede utilizar en una consulta destinada a fuentes de datos. Cuando se procesa la consulta, la indicación de fecha y hora se obtendrá del registro CURRENT TIMESTAMP del servidor federado, no de las fuentes de datos.

Ejemplo: Inserte una fila en la tabla IN_TRAY. El valor de la columna RECEIVED mostrará la indicación de fecha y hora en la que se ha añadido la fila. Los valores de las otras tres columnas se obtienen de las variables del lenguaje principal SRC (char(8)), SUB (char(64)) y TXT (VARCHAR(200)).

```
INSERT INTO IN_TRAY
  VALUES (CURRENT TIMESTAMP, :SRC, :SUB, :TXT)
```

CURRENT TIMEZONE

El registro especial CURRENT TIMEZONE (o CURRENT_TIMEZONE) especifica la diferencia entre UTC (Hora coordinada universal, conocida anteriormente como GMT) y la hora local del servidor de aplicaciones. La diferencia se representa por un período de tiempo (un número decimal cuyos dos primeros dígitos corresponden a las horas, los dos siguientes a los minutos y los dos últimos a los segundos). El número de horas está entre -24 y 24, exclusive. Al restar CURRENT TIMEZONE de la hora local se convierte esa hora a UTC. La hora se calcula a partir de la hora del sistema operativo en el momento en que se ejecuta la sentencia de SQL. (El valor de CURRENT TIMEZONE se determina a partir de las funciones de tiempo de ejecución de C).

El registro especial CURRENT TIMEZONE se puede utilizar allí donde se utilice una expresión de tipo de datos DECIMAL(6,0); por ejemplo, en operaciones aritméticas de hora e indicación de fecha y hora.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT TIMEZONE no se hereda de la sentencia que la invoca.

Ejemplo: Inserte un registro en la tabla IN_TRAY utilizando una indicación de fecha y hora UTC para la columna RECEIVED.

```
INSERT INTO IN_TRAY VALUES (
  CURRENT TIMESTAMP - CURRENT TIMEZONE,
       :source,
       :subject,
       :notetext )
```

CURRENT USER

El registro especial CURRENT USER (o CURRENT_USER) especifica el ID de autorización que se va a utilizar para autorización de sentencia. Para sentencias de SQL estático, el valor representa el ID de autorización que se utilizó cuando se vinculó el paquete. Para sentencias de SQL dinámico, el valor es el mismo que el valor del registro especial SESSION_USER para paquetes vinculados con la opción de vinculación DYNAMICRULES(RUN). El tipo de datos del registro es VARCHAR(128).

Ejemplo: Seleccione nombres de tablas cuyo esquema coincida con el valor del registro especial CURRENT USER.

```
SELECT TABNAME FROM SYSCAT.TABLES
  WHERE TABSCHEMA = CURRENT USER AND TYPE = 'T'
```

Si esta sentencia se ejecuta como una sentencia de SQL estático, devuelve las tablas cuyo nombre de esquema coincide con el vinculador del paquete que incluye la sentencia. Si esta sentencia se ejecuta como una sentencia de SQL dinámico, devuelve las tablas cuyo nombre de esquema coincide con el valor actual del registro especial SESSION_USER.

SESSION_USER

El registro especial SESSION_USER especifica el ID de autorización que se debe utilizar para la sesión actual. El valor de este registro se utiliza para la comprobación de autorización de sentencias de SQL dinámico cuando el comportamiento de ejecución DYNAMICRULES está en vigor para el paquete. El tipo de datos del registro es VARCHAR(128).

El valor inicial de SESSION_USER para una nueva conexión es el mismo que el valor del registro especial SYSTEM_USER. Su valor se puede modificar invocando la sentencia SET SESSION AUTHORIZATION.

SESSION_USER es sinónimo del registro especial USER.

Ejemplo: Determine qué rutinas se pueden ejecutar utilizando SQL dinámico. Supongamos que el comportamiento de ejecución DYNAMICRULES está en vigor para el paquete que emitirá la sentencia de SQL dinámico que invoca la rutina.

```
SELECT SCHEMA, SPECIFICNAME FROM SYSCAT.ROUTINEAUTH
  WHERE GRANTEE = SESSION_USER
  AND EXECUTEAUTH IN ('Y', 'G')
```

Información relacionada:

- "Sentencia SET SESSION AUTHORIZATION" en la publicación *Consulta de SQL, Volumen 2*

SYSTEM_USER

El registro especial SYSTEM_USER especifica el ID de autorización del usuario que se ha conectado a la base de datos. El valor de este registro sólo se puede modificar conectándose como un usuario con otro ID de autorización. El tipo de datos del registro es VARCHAR(128).

Consulte “Ejemplo” en la descripción de la sentencia SET SESSION AUTHORIZATION.

Información relacionada:

- “Sentencia SET SESSION AUTHORIZATION” en la publicación *Consulta de SQL, Volumen 2*

USER

El registro especial USER especifica el ID de autorización de tiempo de ejecución que se pasa al gestor de bases de datos cuando se inicia una aplicación en una base de datos. El tipo de datos del registro es VARCHAR(128).

Cuando se utiliza en una sentencia de SQL incluida en una rutina, USER no se hereda de la sentencia que lo invoca.

Ejemplo: Seleccione todas las notas de la tabla IN_TRAY que el usuario haya colocado ahí.

```
SELECT * FROM IN_TRAY
  WHERE SOURCE = USER
```

Funciones

Una *función de base de datos* es una relación entre un conjunto de valores de datos de entrada y un conjunto de valores del resultado. Por ejemplo, se pueden pasar a la función TIMESTAMP valores de datos de entrada del tipo DATE y TIME y el resultado será TIMESTAMP. Las funciones pueden ser incorporadas o bien definidas por el usuario.

- Con el gestor de bases de datos se proporcionan *funciones incorporadas*. Devuelven un solo valor del resultado y se identifican como parte del esquema SYSIBM. Entre estas funciones se incluyen funciones de columna (por ejemplo, AVG), funciones con operadores (por ejemplo, +) y funciones de conversión (por ejemplo, DECIMAL).
- Las *funciones definidas por el usuario* son funciones que están registradas en una base de datos de SYSCAT.ROUTINES (utilizando la sentencia CREATE FUNCTION). Estas funciones nunca forman parte del esquema SYSIBM. El gestor de bases de datos proporciona un conjunto de estas funciones en un esquema denominado SYSFUN.

  Las funciones definidas por el usuario amplían las posibilidades del sistema de bases de datos añadiendo definiciones de funciones (proporcionadas por usuarios o proveedores) que pueden aplicarse en el propio núcleo de la base de datos. La ampliación de las funciones de la base de datos permite que la base de datos explote las mismas funciones en su núcleo que las que utiliza una aplicación, proporcionando más sinergia entre la aplicación y la base de datos.

Funciones definidas por el usuario externas, de SQL y derivadas

Una función definida por el usuario puede ser una función externa, una función de SQL o una función derivada. Una *función externa* se define en la base de datos con una referencia a una biblioteca de códigos objeto y una función en dicha biblioteca que se ejecutará cuando se invoque la función. Las funciones externas no pueden ser funciones de columna. Una *función de SQL* se define para la base de datos utilizando solamente la sentencia RETURN de SQL. Puede devolver un valor escalar, una fila o una tabla. Las funciones de SQL no pueden ser funciones de columna. Una *función derivada* se define para la base de datos con una referencia a otra función incorporada o definida por el usuario que ya se conoce en la base de datos. Las funciones derivadas pueden ser funciones escalares o funciones de columna. Son útiles para dar soporte a funciones existentes con tipos definidos por el usuario.

Funciones definidas por el usuario escalares, de columna, de fila y de tabla

Cada función definida por el usuario también se clasifica como función escalar, de columna o de tabla. Una *función escalar* es una función que devuelve una respuesta de un solo valor cada vez que se invoca. Por ejemplo, la función incorporada SUBSTR() es una función escalar. Las UDF escalares pueden ser externas o derivadas.

Una *función de columna* es la que recibe un conjunto de valores similares (una columna) y devuelve una respuesta de un solo valor. A veces éstas se denominan también *funciones de agregación* en DB2.Un ejemplo de una función de columna es la función incorporada AVG(). No puede definirse una UDF de columna externa para DB2, pero puede definirse una UDF de columna, que se deriva de las funciones de columna incorporadas. Es útil para tipos diferenciados. Por ejemplo,

si está definido un tipo diferenciado SHOESIZE con un tipo base INTEGER, se podría definir una UDF AVG(SHOESIZE), derivada de la función incorporada AVG(INTEGER), que sería una función de columna.

Una *función de fila* es una función que devuelve una fila de valores. Se puede utilizar sólo como función de transformación, que correlaciona valores de atributos de un tipo estructurado con valores de una fila. Las funciones de fila deben estar definidas como funciones de SQL.

Una *función de tabla* es una función que devuelve una tabla a la sentencia de SQL donde se invoca la función. Sólo se puede hacer referencia a la función en la cláusula FROM de una sentencia SELECT. Una función así puede utilizarse para aplicar la potencia de proceso del lenguaje SQL a datos que no son de DB2, o para convertir tales datos a una tabla DB2. Esta función podría, por ejemplo, tomar un archivo y convertirlo en una tabla, tomar muestras de datos de la Web y disponerlos en forma de tabla o acceder a una base de datos Lotus Notes y devolver información sobre mensajes de correo, tal como la fecha, el remitente y el texto del mensaje. Esta información puede unirse a otras tablas de la base de datos. Una función de tabla se puede definir como una función externa o como una función de SQL. (Una función de tabla no puede ser una función derivada).

Signaturas de función

Una función se identifica por su esquema, un nombre de función el número de parámetros y los tipos de datos de sus parámetros. Esto se denomina *signatura de función*, que debe ser exclusiva en la base de datos. Varias funciones pueden tener el mismo nombre dentro de un esquema, siempre que el número de parámetros o bien los tipos de datos de los parámetros sean diferentes. Un nombre de función para el que existen múltiples instancias de función se llama función *sobrecargada*. Un nombre de función puede estar sobrecargado dentro de un esquema, en cuyo caso hay más de una función con el mismo nombre en el esquema. Estas funciones deben tener tipos de parámetros distintos. Un nombre de función también puede estar sobrecargado en una vía de acceso de SQL, en cuyo caso hay más de una función con el mismo nombre en la vía de acceso. Estas funciones no es necesario que tengan tipos de parámetros distintos.

Una función se puede invocar haciendo referencia (en un contexto que lo permita) a su nombre calificado (esquema y nombre de función), seguido por la lista de argumentos, entre paréntesis. También se puede invocar sin especificar el nombre del esquema, obteniendo como resultado una serie de posibles funciones de diferentes esquemas que tienen los mismos parámetros o bien parámetros aceptables. En este caso, la *vía de acceso de SQL* se utiliza como ayuda para resolver la función. La vía de acceso de SQL es una lista de esquemas que se examinan para identificar una función con el mismo nombre, número de parámetros y tipos de datos aceptables. Para las sentencias de SQL estático, la vía de acceso de SQL se especifica utilizando la opción de vinculación FUNCPATH. Para las sentencias de SQL dinámico, la vía de acceso de SQL es el valor del registro especial CURRENT PATH.

El acceso a las funciones se controla mediante el privilegio EXECUTE. Se utilizan sentencias GRANT y REVOKE para especificar quién puede o no puede ejecutar una función o conjunto de funciones determinadas. Se necesita el privilegio EXECUTE (o la autoridad DBADM) para invocar una función. La persona que define la función recibe el privilegio EXECUTE de forma automática. Si se trata de una función externa o de una función de SQL que tienen la opción WITH GRANT en todos los objetos subyacentes, la persona que la define también recibe la opción

WITH GRANT con el privilegio EXECUTE sobre la función. La persona que la define (o SYSADM o DBADM) debe otorgarlo entonces al usuario que desee invocar la función desde una sentencia de SQL o hacer referencia a la misma en una sentencia de DDL (como, por ejemplo, CREATE VIEW, CREATE TRIGGER o al definir una restricción) o crear otra función derivada de esta función. Si no se otorga a un usuario el privilegio EXECUTE, el algoritmo de resolución de función no tendrá en cuenta la función aunque ésta se corresponda mucho mejor. Las funciones incorporadas (funciones SYSIBM) y las funciones SYSFUN tienen otorgado el privilegio EXECUTE en PUBLIC de forma implícita.

Resolución de función

Después de invocar una función, el gestor de bases de datos debe decidir cuál de las posibles funciones con el mismo nombre es la "más apropiada ". Esto incluye la resolución de funciones incorporadas y definidas por el usuario.

Un *argumento* es un valor que se pasa a una función en una invocación. Cuando se invoca una función en SQL, se pasa una lista de cero o más argumentos. Son argumentos posicionales en tanto que la semántica de dichos argumentos viene determinada por su posición en la lista de argumentos. Un *parámetro* es una definición formal de una entrada para una función. Cuando una función está definida en la base de datos, ya sea internamente (una función incorporada) o por el usuario (una función definida por el usuario), se especifican sus parámetros (cero o más) y el orden de sus definiciones define su posición y su semántica. Por lo tanto, cada parámetro es una entrada posicional particular de una función. En la invocación, un argumento corresponde a un parámetro determinado en virtud a la posición que éste ocupe en la lista de argumentos.

El gestor de bases de datos utiliza el nombre de la función que se facilita en la invocación, el privilegio EXECUTE sobre la función, el número y los tipos de datos de los argumentos, todas las funciones que tienen el mismo nombre en la vía de acceso de SQL y los tipos de datos de sus parámetros correspondientes como punto de referencia para decidir si se selecciona o no una función. A continuación se muestran los resultados posibles del proceso de decisión:

- Una función determinada se considera como la mejor. Por ejemplo, con las funciones denominadas RISK en el esquema TEST con las signaturas definidas como:

  ```
  TEST.RISK(INTEGER)
  TEST.RISK(DOUBLE)
  ```

  una vía de acceso de SQL que incluya el esquema TEST y la siguiente referencia de función (donde DB es una columna DOUBLE):

  ```
  SELECT ... RISK(DB) ...
  ```

  se elegirá el segundo RISK.

  La siguiente referencia de función (donde SI es una columna SMALLINT):

  ```
  SELECT ... RISK(SI) ...
  ```

  elegirá el primer RISK, ya que SMALLINT se puede promover a INTEGER y es una coincidencia mejor que DOUBLE, que se encuentra más abajo en la lista de prioridad.

  Cuando se tienen en cuenta argumentos que son tipos estructurados, la lista de prioridad incluye los supertipos del tipo estático del argumento. La función que mejor se ajusta es la definida con el parámetro de supertipo más cercano, en la jerarquía de tipos estructurados, al tipo estático del argumento de función.

- Ninguna función se considera la mejor. Tomando como ejemplo las dos mismas funciones del caso anterior y la siguiente referencia de función (donde C es una columna CHAR(5)):

  ```
  SELECT ... RISK(C) ...
  ```

  el argumento es incoherente con el parámetro de las dos funciones RISK.
- Una función determinada se selecciona según la vía de acceso de SQL y el número de argumentos, así como sus tipos de datos, que se han pasado en la invocación. Por ejemplo, dadas unas funciones denominadas RANDOM con las signaturas definidas como:

  ```
  TEST.RANDOM(INTEGER)
  PROD.RANDOM(INTEGER)
  ```

  y una vía de acceso de SQL de:

  ```
  "TEST","PROD"
  ```

  la siguiente referencia de función:

  ```
  SELECT ... RANDOM(432) ...
  ```

  elegirá TEST.RANDOM, ya que las dos funciones RANDOM son coincidencias igualmente buenas (coincidencias exactas en este caso particular) y ambos esquemas están en la vía de acceso, pero TEST precede a PROD en la vía de acceso de SQL.

Determinación de la mejor opción

La comparación de los tipos de datos de los argumentos con los tipos de datos definidos de los parámetros de las funciones en cuestión, constituye la base primordial para tomar la decisión de qué función de un grupo de funciones con el mismo nombre se considera “más adecuada”. Tenga en cuenta que los tipos de datos de los resultados de las funciones o el tipo de función (de columna, escalar o de tabla) en cuestión no se tiene en cuenta en esta determinación.

La resolución de las funciones se realiza siguiendo los pasos siguientes:

1. En primer lugar, busque todas aquellas funciones del catálogo (SYSCAT.ROUTINES) y funciones incorporadas que cumplan las condiciones siguientes:
   - En las invocaciones donde se ha especificado el nombre de esquema (una referencia calificada), el nombre de esquema y el nombre de función coinciden con el nombre de invocación.
   - En las invocaciones donde no se ha especificado el nombre de esquema (una referencia no calificada), el nombre de función coincide con el nombre de invocación y tiene un nombre de esquema que coincide con uno de los esquemas de la vía de acceso de SQL.
   - La persona que la invoca tiene el privilegio EXECUTE sobre la función.
   - El número de parámetros definidos debe coincidir con la invocación.
   - Cada argumento de invocación coincide con el tipo de datos del parámetro definido correspondiente de la función o es “promocionable” al mismo.
2. A continuación, examine de izquierda a derecha cada argumento de la invocación de la función. Para cada argumento, elimine todas las funciones que no sean la mejor coincidencia para ese argumento. La mejor opción para un argumento dado es el primer tipo de datos que aparece en la lista de prioridad correspondiente al tipo de datos del argumento para el cual existe una función con un parámetro de ese tipo de datos. En esta comparación no se tienen en cuenta las longitudes, precisiones y escalas ni el atributo FOR BIT DATA. Por

ejemplo, un argumento DECIMAL(9,1) se considera una coincidencia exacta para un parámetro DECIMAL(6,5) mientras que un argumento VARCHAR(19) es una coincidencia exacta para un parámetro VARCHAR(6).

La mejor coincidencia para un argumento de tipo estructurado definido por el usuario es el propio argumento; la siguiente mejor coincidencia es el supertipo inmediato, y así sucesivamente para cada supertipo del argumento. Observe que sólo se tiene en cuenta el tipo estático (tipo declarado) del argumento de tipo estructurado, no el tipo dinámico (tipo más específico).

3. Si después del paso 2 queda más de una función elegible, todas las funciones elegibles restantes deben tener signaturas idénticas pero encontrarse en esquemas diferentes. Elija la función cuyo esquema aparezca antes en la vía de acceso de SQL del usuario.
4. Si después del paso 2 no queda ninguna función elegible, se devolverá un error (SQLSTATE 42884).

Consideraciones sobre la vía de acceso de funciones para las funciones incorporadas

Las funciones incorporadas residen en un esquema especial denominado SYSIBM. Hay funciones adicionales disponibles en los esquemas SYSFUN y SYSPROC que, sin embargo, no se consideran funciones incorporadas porque se han desarrollado como funciones definidas por el usuario y carecen de consideraciones de proceso especiales. Los usuarios no pueden definir funciones adicionales en los esquemas SYSIBM, SYSFUN ni SYSPROC (ni en ningún otro esquema cuyo nombre empiece por las letras 'SYS').

Como ya se ha indicado, las funciones incorporadas participan en el proceso de resolución de las funciones exactamente como lo hacen las funciones definidas por el usuario. Una diferencia entre ambas, desde el punto de vista de la resolución de función, es que las funciones incorporadas siempre deben tenerse en cuenta durante la resolución de función. Por este motivo, si se omite SYSIBM de los resultados de vía de acceso se asume (para la resolución de las funciones y los tipos de datos) que SYSIBM es el primer esquema de la vía de acceso.

Por ejemplo, si la vía de acceso de SQL de un usuario está definida de la siguiente manera:

```
"SHAREFUN","SYSIBM","SYSFUN"
```

y hay una función LENGTH definida en el esquema SHAREFUN con el mismo número y los mismos tipos de argumentos que SYSIBM.LENGTH, una referencia no calificada a LENGTH en la sentencia de SQL de este usuario hará que se seleccione SHAREFUN.LENGTH. No obstante, si la vía de acceso de SQL del usuario está definida de la siguiente forma:

```
"SHAREFUN","SYSFUN"
```

y existe la misma función SHAREFUN.LENGTH, una referencia no calificada a LENGTH en la sentencia de SQL de este usuario hará que se seleccione SYSIBM.LENGTH, ya que SYSIBM aparece implícitamente antes en la vía de acceso.

Para minimizar los posibles problemas en este área:

- No utilice nunca los nombres de funciones incorporadas para funciones definidas por el usuario.

- Si, por algún motivo, es necesario crear una función definida por el usuario con el mismo nombre que una función incorporada, asegúrese de calificar todas las referencias a la misma.

Ejemplo de resolución de función

A continuación se muestra un ejemplo de una resolución de función satisfactoria. (Observe que no se muestran todas las palabras clave necesarias.)

Existen siete funciones ACT, en tres esquemas diferentes, registradas del modo siguiente:

```
CREATE FUNCTION AUGUSTUS.ACT (CHAR(5), INT, DOUBLE)  SPECIFIC ACT_1 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_2 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE, INT) SPECIFIC ACT_3 ...
CREATE FUNCTION JULIUS.ACT (INT, DOUBLE, DOUBLE) SPECIFIC ACT_4 ...
CREATE FUNCTION JULIUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_5 ...
CREATE FUNCTION JULIUS.ACT (SMALLINT, INT, DOUBLE) SPECIFIC ACT_6 ...
CREATE FUNCTION NERO.ACT (INT, INT, DEC(7,2)) SPECIFIC ACT_7 ...
```

La referencia de función es la siguiente (donde I1 e I2 son columnas INTEGER y D es una columna DECIMAL):

```
SELECT ... ACT(I1, I2, D) ...
```

Supongamos que la aplicación que efectúa esta referencia tiene una vía de acceso de SQL establecida como:

```
"JULIUS","AUGUSTUS","CAESAR"
```

De acuerdo con el algoritmo...

- La función con el nombre específico ACT_7 se elimina como candidato, porque el esquema NERO no está incluido en la vía de acceso de SQL.
- La función con el nombre específico ACT_3 se elimina como candidato, porque tiene el número incorrecto de parámetros. ACT_1 y ACT_6 se eliminan porque, en ambos casos, el primer argumento no se puede promocionar al tipo de datos del primer parámetro.
- Como sigue habiendo más de un candidato, los argumentos se tienen en cuenta siguiendo un orden.
- Para el primer argumento, las funciones restantes, ACT_2, ACT_4 y ACT_5 coinciden exactamente con el tipo de argumento. No se puede pasar por alto ninguna de las funciones; así pues, se debe examinar el argumento siguiente.
- Para este segundo argumento, ACT_2 y ACT_5 coinciden exactamente, pero no sucede lo mismo con ACT_4, por lo cual queda descartado. Se examina el siguiente argumento para determinar alguna diferencia entre ACT_2 y ACT_5.
- Para el tercer y último argumento, ni ACT_2 ni ACT_5 coinciden exactamente con el tipo de argumento, pero ambos son igualmente aceptables.
- Quedan dos funciones, ACT_2 y ACT_5, con signaturas de parámetros idénticas. La criba final consiste en determinar el esquema de qué función aparece primero en la vía de acceso de SQL y, en base a ello, se elige ACT_5.

invocación de funciones

Cuando ya se ha seleccionado la función, pueden darse todavía algunos motivos por los cuales no se pueda utilizar alguna función. Cada función está definida para devolver un resultado con un tipo de datos concreto. Si este tipo de datos resultante no es compatible con el contexto en el que se invoca la función, se producirá un error. Por ejemplo, con las funciones denominadas STEP, en esta ocasión con tipos de datos diferentes como resultado:

```
STEP(SMALLINT) devuelve CHAR(5)
STEP(DOUBLE)   devuelve INTEGER
```

y la referencia de función siguiente (donde S es una columna SMALLINT):

```
SELECT ... 3 + STEP(S) ...
```

como hay una coincidencia exacta de tipo de argumento, se elige la primera operación STEP. Se produce un error en la sentencia porque el tipo resultante es CHAR(5) en lugar de un tipo numérico, tal como necesita un argumento del operador de suma.

Otros ejemplos en los que puede ocurrir esto son los siguientes, ambos darán como resultado un error en la sentencia:

- Se ha hecho referencia a la función en una cláusula FROM, pero la función seleccionada en el paso de resolución de las funciones ha sido una función escalar o de columna.
- El caso contrario en el que el contexto llama a una función escalar o de columna y la resolución de la función selecciona una función de tabla.

En los casos donde los argumentos de la invocación de la función no coinciden exactamente con los tipos de datos de los parámetros de la función seleccionada, los argumentos se convierten al tipo de datos del parámetro durante la ejecución, utilizando las mismas reglas que para la asignación a columnas. También se incluye el caso en el que la precisión, escala o longitud difiere entre el argumento y el parámetro.

Semántica de vinculación conservadora

Existen casos en los que las rutinas y los tipos de datos se resuelven cuando se procesa una sentencia y el gestor de bases de datos debe poder repetir esta resolución. Esto sucede en:

- Sentencias en paquetes de DML estático
- Vistas
- Activadores
- Restricciones de comprobación
- Rutinas de SQL

En el caso de las sentencias en paquetes de DML estático, las referencias a la rutina y al tipo de datos se resuelven durante una operación de vinculación. Las referencias a la rutina y al tipo de datos en las vistas, activadores, rutinas de SQL y restricciones de comprobación se resuelven cuando se crea el objeto de base de datos.

Si la resolución de la rutina se realiza de nuevo en alguna referencia a rutina de estos objetos, podría cambiar de comportamiento si:

- Se ha añadido una rutina nueva con una signatura más adecuada pero el ejecutable real realiza operaciones distintas.
- A la persona que la ha definido se le ha otorgado el privilegio de ejecución sobre una rutina con una signatura más adecuada pero el ejecutable real realiza operaciones distintas.

Similarmente, si la resolución se ejecuta de nuevo para cualquier tipo de datos de estos objetos, podría cambiar el comportamiento si se ha añadido un nuevo tipo de datos con el mismo nombre en un esquema diferente que también está en la vía de acceso de SQL. Para evitar esto, el gestor de bases de datos aplica la *semántica de*

vinculación conservadora cuando lo considera necesario. Esto asegura que las referencias a la rutina y al tipo de datos se resuelvan utilizando la misma vía de acceso de SQL y el mismo conjunto de rutinas con las que se resolvió cuando se vinculó anteriormente. La indicación de fecha y hora de creación de las rutinas y los tipos de datos tenidos en cuenta durante la resolución no es posterior a la hora en que se vinculó la sentencia. (Las funciones incorporadas añadidas a partir de la Versión 6.1 tienen una indicación de fecha y hora de creación basada en la hora de creación o migración de la base de datos.) De esta forma, sólo se tendrán en cuenta las rutinas y los tipos de datos que se tuvieron en cuenta durante la resolución de la rutina y del tipo de datos cuando se procesó originalmente la sentencia. Por tanto, las rutinas y los tipos de datos creados o autorizados recientemente no se tendrán en cuenta cuando se aplica la semántica de vinculación conservadora.

Piense en una base de datos con dos funciones que tienen las signaturas SCHEMA1.BAR(INTEGER) y SCHEMA2.BAR(DOUBLE). Supongamos que una vía de acceso de SQL contiene los dos esquemas, SCHEMA1 y SCHEMA2 (aunque su orden en la vía de acceso de SQL carece de importancia). A USER1 se le ha otorgado el privilegio EXECUTE sobre la función SCHEMA2.BAR(DOUBLE). Supongamos que USER1 crea una vista que llama a BAR(INT_VAL). Esta se resolverá en la función SCHEMA2.BAR(DOUBLE). La vista siempre utilizará SCHEMA2.BAR(DOUBLE), aunque alguien otorgue a USER1 el privilegio EXECUTE sobre SCHEMA1.BAR(INTEGER) después de crear la vista.

En el caso de los paquetes de DML estático, los paquetes se pueden volver a vincular implícitamente o emitiendo explícitamente el mandato REBIND (o la API correspondiente) o el mandato BIND (o la API correspondiente). La revinculación implícita se ejecuta siempre para resolver rutinas y tipos de datos con la semántica de vinculación conservadora. El mandato REBIND proporciona la posibilidad de resolver con la semántica de vinculación conservadora (RESOLVE CONSERVATIVE) o resolver teniendo en cuenta las rutinas y tipos datos nuevos (RESOLVE ANY, la opción por omisión).

La revinculación implícita de un paquete siempre resuelve la misma rutina. Aunque se hay otorgado el privilegio EXECUTE sobre una rutina más adecuada, la rutina no se tendrá en cuenta. La revinculación explícita de un paquete puede dar como resultado la selección de una rutina distinta. (Pero si se especifica RESOLVE CONSERVATIVE, la resolución de la rutina seguirá la semántica de vinculación conservadora).

Si una rutina se ha especificado durante la creación de una vista, activador, restricción o cuerpo de rutina SQL, la instancia concreta de la rutina que se utilizará viene determinada por la resolución de rutina en el momento en que se crea el objeto. Aunque posteriormente se otorgue el privilegio EXECUTE después de crear el objeto, la rutina concreta que el objeto utiliza no cambiará.

Piense en una base de datos con dos funciones que tienen las signaturas SCHEMA1.BAR(INTEGER) y SCHEMA2.BAR(DOUBLE). A USER1 se le ha otorgado el privilegio EXECUTE sobre la función SCHEMA2.BAR(DOUBLE). Supongamos que USER1 crea una vista que llama a BAR(INT_VAL). Esta se resolverá en la función SCHEMA2.BAR(DOUBLE). La vista siempre utilizará SCHEMA2.BAR(DOUBLE), aunque alguien otorgue a USER1 el privilegio EXECUTE sobre SCHEMA1.BAR(INTEGER) después de crear la vista.

El mismo comportamiento se produce en otros objetos de la base de datos. Por ejemplo, si un paquete se revincula de forma implícita (tal vez después de eliminar un índice), el paquete hará referencia a la misma rutina concreta tanto antes como

después de la revinculación implícita. Sin embargo, la revinculación *explícito* de un paquete, puede dar como resultado la selección de una rutina distinta.

Información relacionada:

- "CURRENT PATH" en la página 152
- "Promoción de tipos de datos" en la página 105
- "Asignaciones y comparaciones" en la página 110

Métodos

Un método de base de datos de un tipo estructurado es una relación entre un conjunto de valores de datos de entrada y un conjunto de valores resultantes, donde el primer valor de entrada (o *argumento sujeto*) tiene el mismo tipo, o es un subtipo del tipo sujeto (también llamado *parámetro sujeto*) del método. Por ejemplo, es posible pasar valores de datos de entrada de tipo VARCHAR a un método denominado CITY, de tipo ADDRESS y el resultado será un valor ADDRESS (o un subtipo de ADDRESS).

Los métodos se definen implícita o explícitamente, como parte de la definición de un tipo estructurado definido por el usuario.

Los métodos definidos implícitamente se crean para cada tipo estructurado. Se definen *métodos observadores* para cada atributo del tipo estructurado. Los métodos observadores permiten que una aplicación obtenga el valor de un atributo para una instancia del tipo. También se definen *métodos mutadores* para cada atributo, que permiten que una aplicación cambie la instancia de tipo modificando el valor de un atributo de una instancia de tipo. El método CITY descrito anteriormente es un ejemplo de método mutador para el tipo ADDRESS.

Los métodos definidos explícitamente o *métodos definidos por el usuario* son métodos que se registran en el catálogo SYSCAT.ROUTINES de una base de datos, utilizando una combinación de las sentencias CREATE TYPE (o ALTER TYPE ADD METHOD) y CREATE METHOD. Todos los métodos definidos para un tipo estructurado se definen en el mismo esquema que el tipo.

Los métodos definidos por el usuario para tipos estructurados amplían la función de sistema de bases de datos añadiendo definiciones de método (proporcionadas por usuarios o proveedores) que pueden aplicarse a instancias de tipo estructurado en el núcleo de la base de datos. La definición de los métodos de la base de datos permite que la base de datos explote los mismos métodos en su núcleo que los que utiliza una aplicación, proporcionando más sinergia entre la aplicación y la base de datos.

Métodos definidos por el usuario externos y SQL

Un método definido por el usuario puede ser externo o estar basado en una expresión SQL. Un método externo se define para la base de datos con una referencia a una biblioteca de códigos objeto y una función en dicha biblioteca que se ejecutará cuando se invoque el método. Un método basado en una expresión SQL devuelve el resultado de la expresión SQL cuando se invoca el método. Tales métodos no necesitan ninguna biblioteca de códigos objeto, ya que están escritos completamente en SQL.

Un método definido por el usuario puede devolver un solo valor cada vez que se invoca. Este valor puede ser un tipo estructurado. Un método se puede definir como *preservador del tipo* (utilizando SELF AS RESULT), para permitir que el tipo dinámico del argumento sujeto sea el tipo devuelto del método. Todos los métodos mutadores definidos implícitamente son preservadores del tipo.

Signaturas de método

Un método se identifica por su tipo sujeto, un nombre de método, el número de parámetros y los tipos de datos de sus parámetros. Esto se denomina una *signatura de método* y debe ser exclusiva en la base de datos.

Puede existir más de un método con el mismo nombre para un tipo estructurado, siempre que:

- El número de parámetros o los tipos de datos de los parámetros sean diferentes o
- Los métodos formen parte de la misma jerarquía de métodos (es decir, los métodos estén en una relación de alteración temporal o alteren temporalmente el mismo método original) o
- No exista la misma signatura de función (utilizando el tipo sujeto o cualquiera de sus subtipos o supertipos como primer parámetro).

Un nombre de método que tiene varias instancias de método se denomina *método sobrecargado*. Un nombre de método puede estar sobrecargado dentro de un tipo, lo que significa que existe más de un método de ese nombre para el tipo (todos los cuales tienen diferentes tipos de parámetros). Un nombre de método también puede estar sobrecargado en la jerarquía de tipos sujeto, en cuyo caso existe más de un método con ese nombre en la jerarquía de tipos. Estos métodos deben tener tipos de parámetros distintos.

Un método se puede invocar haciendo referencia (en un contexto permitido) al nombre de método, precedido por una referencia a una instancia de tipo estructurado (el argumento sujeto) y por el operador de doble punto. A continuación debe seguir una lista de argumentos entre paréntesis. El método que se invoca realmente depende del tipo estático del tipo sujeto, utilizando el proceso de resolución de métodos descrito en la sección siguiente. Los métodos definidos con WITH FUNCTION ACCESS también se pueden invocar utilizando la invocación de funciones, en cuyo caso se aplican las reglas normales para la resolución de la función.

Si la resolución de la función da como resultado un método definido con WITH FUNCTION ACCESS, se procesan todos los pasos siguientes de invocación de métodos.

El acceso a los métodos se controla mediante el privilegio EXECUTE. Se utilizan sentencias GRANT y REVOKE para especificar quién puede o no puede ejecutar un método o conjunto de métodos determinado. Se necesita el privilegio EXECUTE (o la autoridad DBADM) para invocar un método. La persona que define el método recibe el privilegio EXECUTE de forma automática. Si se trata de un método externo o un método SQL que tienen la opción WITH GRANT en todos los objetos subyacentes, la persona que lo define también recibe la opción WITH GRANT con el privilegio EXECUTE sobre el método. La persona que lo define (o SYSADM o DBADM) debe otorgarlo entonces al usuario que desee invocar el método desde una sentencia de SQL o hacer referencia al mismo en una sentencia de DDL (como, por ejemplo, CREATE VIEW, CREATE TRIGGER o al definir una restricción). Si no se otorga a un usuario el privilegio EXECUTE, el algoritmo de resolución de métodos no tendrá en cuenta el método aunque éste se corresponda mucho mejor.

Resolución de métodos

Después de invocar un método, el gestor de bases de datos debe decidir cuál de los posibles métodos con el mismo nombre es el “más apropiado ”. Las funciones (incorporadas o definidas por el usuario) no se tienen en cuenta durante la resolución del método.

Un *argumento* es un valor que se pasa a un método en una invocación. Cuando un método se invoca en SQL, se le pasa el argumento sujeto (de algún tipo

estructurado) y opcionalmente una lista de argumentos. Son argumentos posicionales en tanto que la semántica de dichos argumentos viene determinada por su posición en la lista de argumentos. Un *parámetro* es una definición formal de una entrada en un método. Cuando se define un método para la base de datos, ya sea implícitamente (método generado por el sistema para un tipo) o por un usuario (método definido por el usuario), se especifican sus parámetros (con el parámetro sujeto como primer parámetro) y el orden de sus definiciones determina sus posiciones y su semántica. Por tanto, cada parámetro es una entrada posicional determinada de un método. En la invocación, un argumento corresponde a un parámetro determinado en virtud a la posición que éste ocupe en la lista de argumentos.

El gestor de bases de datos utiliza el nombre de método proporcionado en la invocación, el privilegio EXECUTE sobre el método, el número y los tipos de datos de los argumentos, todos los métodos que tienen el mismo nombre para el tipo estático del argumento sujeto y los tipos de datos de sus parámetros correspondientes como base para decidir si selecciona o no un método. A continuación se muestran los resultados posibles del proceso de decisión:

- Un método determinado se considera que es el más apropiado. Por ejemplo, para los métodos denominados RISK del tipo SITE con signaturas definidas como:

  ```
  PROXIMITY(INTEGER) FOR SITE
  PROXIMITY(DOUBLE) FOR SITE
  ```

  la siguiente invocación de método (donde ST es una columna SITE, DB es una columna DOUBLE):

  ```
  SELECT ST..PROXIMITY(DB) ...
  ```

  se elegiría el segundo PROXIMITY.

  La siguiente invocación de método (donde SI es una columna SMALLINT):

  ```
  SELECT ST..PROXIMITY(SI) ...
  ```

  elegirá el primer PROXIMITY, ya que SMALLINT se puede promover a INTEGER y es una coincidencia mejor que DOUBLE, que se encuentra más abajo en la lista de prioridad.

  Cuando se tienen en cuenta argumentos que son tipos estructurados, la lista de prioridad incluye los supertipos del tipo estático del argumento. La función que mejor se ajusta es la definida con el parámetro de supertipo más cercano, en la jerarquía de tipos estructurados, al tipo estático del argumento de función.
- Ningún método se considera una opción aceptable. Tomando como ejemplo las dos mismas funciones del caso anterior y la siguiente referencia de función (donde C es una columna CHAR(5)):

  ```
  SELECT ST..PROXIMITY(C) ...
  ```

  el argumento es incoherente con el parámetro de las dos funciones PROXIMITY.
- Se selecciona un método determinado basándose en los métodos de la jerarquía de tipos y en el número y tipos de datos de los argumentos pasados en la invocación. Por ejemplo, para los métodos denominados RISK de los tipos SITE y DRILLSITE (un subtipo de SITE) con signaturas definidas como:

  ```
  RISK(INTEGER) FOR DRILLSITE
  RISK(DOUBLE) FOR SITE
  ```

  y la siguiente invocación de método (donde DRST es una columna DRILLSITE, DB es una columna DOUBLE):

```
SELECT DRST..RISK(DB) ...
```

se elegirá el segundo RISK, ya que DRILLSITE se puede promocionar a SITE.

La siguiente referencia a método (donde SI es una columna SMALLINT):

```
SELECT DRST..RISK(SI) ...
```

elegirá el primer RISK, ya que SMALLINT se puede promocionar a INTEGER, que está más cerca en la lista de prioridad que DOUBLE, y DRILLSITE es una opción mejor que SITE, que es un supertipo.

Los métodos con la misma jerarquía de tipos no pueden tener las mismas signaturas, teniendo en cuenta parámetros distintos al parámetro sujeto.

Determinación de la mejor opción

La comparación de los tipos de datos de los argumentos con los tipos de datos definidos de los parámetros de los métodos en cuestión, constituye la base primordial para decidir qué método de un grupo de métodos con el mismo nombre se considera el "más apropiado". Observe que los tipos de datos de los resultados de los métodos en cuestión no intervienen en esa decisión.

La resolución del método se realiza siguiendo los pasos siguientes:

1. En primer lugar, busque todos los métodos del catálogo (SYSCAT.ROUTINES) que cumplan las condiciones siguientes:
   - El nombre del método coincide con el nombre de invocación, y el parámetro sujeto es el mismo tipo o es un supertipo del tipo estático del argumento sujeto.
   - La persona que lo invoca tiene el privilegio EXECUTE sobre el método.
   - El número de parámetros definidos debe coincidir con la invocación.
   - Cada argumento de invocación coincide con el tipo de datos del parámetro definido correspondiente del método o es "promocionable" a ese tipo.
2. A continuación, examine de izquierda a derecha cada argumento de la invocación del método. El argumento situado más a la izquierda (y por tanto el primer argumento) es el parámetro SELF implícito. Por ejemplo, un método definido para el tipo ADDRESS_T tiene un primer parámetro implícito de tipo ADDRESS_T. Para cada argumento, elimine todas las funciones que no sean la mejor coincidencia para ese argumento. La mejor opción para un argumento dado es el primer tipo de datos que aparece en la lista de prioridad correspondiente al tipo de datos del argumento para el cual existe una función con un parámetro de ese tipo de datos. La longitud, la precisión, la escala y el atributo FOR BIT DATA no se tienen en cuenta en esta comparación. Por ejemplo, un argumento DECIMAL(9,1) se considera una coincidencia exacta para un parámetro DECIMAL(6,5) mientras que un argumento VARCHAR(19) es una coincidencia exacta para un parámetro VARCHAR(6).

   La mejor coincidencia para un argumento de tipo estructurado definido por el usuario es el propio argumento; la siguiente mejor coincidencia es el supertipo inmediato, y así sucesivamente para cada supertipo del argumento. Observe que sólo se tiene en cuenta el tipo estático (tipo declarado) del argumento de tipo estructurado, no el tipo dinámico (tipo más específico).
3. Como máximo, después del paso 2 queda un método elegible. Este es el método que se elige.
4. Si después del paso 2 no queda ningún método elegible, se produce un error (SQLSTATE 42884).

Ejemplo de resolución de método

A continuación se muestra un ejemplo de una resolución de método satisfactoria.

Existen siete métodos FOO para tres tipos estructurados definidos en una jerarquía de GOVERNOR como un subtipo de EMPEROR, como un subtipo de HEADOFSTATE, registrados con las signaturas siguientes:

```
CREATE METHOD FOO (CHAR(5), INT, DOUBLE)  FOR HEADOFSTATE SPECIFIC FOO_1 ...
CREATE METHOD FOO (INT, INT, DOUBLE)      FOR HEADOFSTATE SPECIFIC FOO_2 ...
CREATE METHOD FOO (INT, INT, DOUBLE, INT) FOR HEADOFSTATE SPECIFIC FOO_3 ...
CREATE METHOD FOO (INT, DOUBLE, DOUBLE)   FOR EMPEROR     SPECIFIC FOO_4 ...
CREATE METHOD FOO (INT, INT, DOUBLE)      FOR EMPEROR     SPECIFIC FOO_5 ...
CREATE METHOD FOO (SMALLINT, INT, DOUBLE) FOR EMPEROR     SPECIFIC FOO_6 ...
CREATE METHOD FOO (INT, INT, DEC(7,2))    FOR GOVERNOR    SPECIFIC FOO_7 ...
```

La referencia al método es la siguiente (donde I1 e I2 son columnas INTEGER, D es una columna DECIMAL y E es una columna EMPEROR):

```
SELECT E..FOO(I1, I2, D) ...
```

De acuerdo con el algoritmo...

- FOO_7 se elimina como candidato, porque el tipo GOVERNOR es un subtipo (no un supertipo) de EMPEROR.
- FOO_3 se elimina como candidato, porque tiene un número erróneo de parámetros.
- FOO_1 y FOO_6 se eliminan porque, en ambos casos, el primer argumento (no el argumento sujeto) no se puede promocionar al tipo de datos del primer parámetro. Como sigue habiendo más de un candidato, los argumentos se tienen en cuenta siguiendo un orden.
- Para el argumento sujeto, FOO_2 es un supertipo, mientras que FOO_4 y FOO_5 coinciden con el argumento sujeto.
- Para el primer argumento, los métodos restantes, FOO_4 y FOO_5, coinciden exactamente con el tipo del argumento. No se puede asignar ningún método y, por tanto, se debe examinar el argumento siguiente.
- Para este segundo argumento, FOO_5 es una coincidencia exacta pero FOO_4 no lo es, por lo cual se descarta. Esto deja FOO_5 como método elegido.

Invocación de métodos

Una vez seleccionado el método, pueden todavía existir algunos motivos por los cuales no se pueda utilizar el método.

Cada método está definido para devolver un resultado con un tipo de datos específico. Si este tipo de datos resultante no es compatible con el contexto donde se invoca el método, se produce un error. Por ejemplo, supongamos que se definen los siguientes métodos llamados STEP, cada uno con un tipo de datos diferentes como resultado:

```
STEP(SMALLINT) FOR TYPEA RETURNS CHAR(5)
STEP(DOUBLE) FOR TYPEA RETURNS INTEGER
```

y la siguiente referencia a método (donde S es una columna SMALLINT y TA es una columna TYPEA):

```
SELECT 3 + TA..STEP(S) ...
```

en este caso se elige el primer STEP, pues hay una coincidencia exacta del tipo del argumento. Se produce un error en la sentencia, porque el tipo resultante es CHAR(5) en lugar de un tipo numérico, tal como necesita un argumento del operador de suma.

Empezando por el método que se ha seleccionado, se utiliza el algoritmo descrito en "Asignación dinámica de métodos" para crear el conjunto de métodos asignables durante la compilación. En "Asignación dinámica de métodos" se describe con exactitud el método que se invoca.

Observe que cuando el método seleccionado es un método conservador del tipo:

- el tipo estático resultante tras la resolución de la función es el mismo que el tipo estático del argumento sujeto de la invocación del método
- el tipo dinámico resultante cuando se invoca el método es el mismo que el tipo dinámico del argumento sujeto de la invocación del método.

Esto puede ser un subtipo del tipo resultante especificado en la definición del método conservador del tipo, que a su vez puede ser un supertipo del tipo dinámico devuelto realmente cuando se procesa el método.

En los casos donde los argumentos de la invocación del método no coinciden exactamente con los tipos de datos de los parámetros del método seleccionado, los argumentos se convierten al tipo de datos del parámetro durante la ejecución, utilizando las mismas reglas que para la asignación a columnas. Esto incluye el caso en el que la precisión, escala o longitud difiere entre el argumento y el parámetro, pero excluye el caso en el que el tipo dinámico del argumento es un subtipo del tipo estático del parámetro.

Asignación dinámica de métodos

Los métodos proporcionan la funcionalidad y encapsulan los datos de un tipo. Un método se define para un tipo y siempre puede asociarse con este tipo. Uno de los parámetros del método es el parámetro implícito SELF. El parámetro SELF es del tipo para el que se ha declarado el método. El argumento que se pasa al argumento SELF cuando se invoca el método en una sentencia DML se denomina *sujeto*.

Cuando se selecciona un método utilizando la resolución de métodos (vea "Resolución de métodos" en la página 174), o se ha especificado un método en una sentencia de DDL, este método se conoce como el "método autorizado aplicable más específico". Si el sujeto es de tipo estructurado, es posible que el método tenga uno o varios métodos alternativos. Entonces, DB2 debe determinar a cuál de estos métodos debe invocar, en base al tipo dinámico (tipo más específico) del sujeto en tiempo de ejecución. Esta determinación se denomina "determinación del método asignable más específico". Este proceso se describe aquí.

1. En la jerarquía de métodos, busque el método original del que forme parte el método autorizado más específico. Se denomina el *método raíz*.
2. Cree el conjunto de métodos asignables, que debe incluir los siguientes:
   - El método autorizado aplicable más específico.
   - Cualquier método que altere temporalmente el método autorizado aplicable más específico y que esté definido para un tipo que sea un subtipo del sujeto de esta invocación.
3. Determine el método asignable más específico, de la forma siguiente:

a. Empiece con un método arbitrario que sea un elemento del conjunto de métodos asignables y que sea un método del tipo dinámico del sujeto o de uno de sus supertipos. Es el método asignable inicial más específico.
b. Itere por los elementos del conjunto de métodos asignables. Para cada método: Si el método está definido para uno de los subtipos adecuados del tipo para el que está definido el método asignable más específico y si está definido para uno de los supertipos del tipo más específico del sujeto, repita el paso 2 con este método como el método asignable más específico; de lo contrario, siga iterando.

4. Invoque el método asignable más específico.

Ejemplo:

Se proporcionan tres tipos: ″Persona″, ″Empleado″ y″Director″. Existe un método original ″ingresos″, definido para ″Persona″, que calcula los ingresos de una persona. Por omisión, una persona es un desempleado (un niño, un jubilado, etc.). Por lo tanto, ″ingresos″ para el tipo ″Persona″ siempre devuelve cero. Para el tipo ″Empleado″ y para el tipo ″Director″, deben aplicarse algoritmos distintos para calcular los ingresos. Por lo tanto, el método ″ingresos″ para el tipo ″Persona″ se altera temporalmente en ″Empleado″ y en ″Director″.

Cree y rellene una tabla de la manera siguiente:

```
CREATE TABLE aTable (id integer, personColumn Person);
  INSERT INTO aTable VALUES (0, Persona()), (1, Empleado()), (2, Director());
```

Liste todas las personas que tengan unos ingresos mínimos de $40000:

```
SELECT id, persona, name
  FROM aTable
  WHERE persona..ingresos() >= 40000;
```

Utilizando la resolución de métodos, se selecciona el método ″ingresos″ para el tipo ″Persona″ como el método autorizado aplicable más específico.

1. El método raíz es ″ingresos″ para ″Persona″.
2. El segundo paso del algoritmo anterior se lleva a cabo para construir el conjunto de métodos asignables:
   - Se incluye el método ″ingresos″ para el tipo ″Persona″, porque es el método autorizado aplicable más específico.
   - Se incluye el método ″ingresos″ para el tipo ″Empleado″ e ″ingresos″ para ″Director″, porque ambos métodos alteran temporalmente el método raíz y tanto ″Empleado″ como ″Director″ son subtipos de ″Persona″.

   Por lo tanto, el conjunto de métodos asignables es: {″ingresos″ para ″Persona″, ″ingresos″ para ″Empleado″, ″ingresos″ para ″Director″}.
3. Determine el método asignable más específico:
   - Para un sujeto cuyo tipo más específico sea ″Persona″:
     a. Deje que el método asignable inicial más específico sea ″ingresos″ para el tipo ″Persona″.
     b. Como no hay ningún otro método en el conjunto de métodos asignables que esté definido para un subtipo adecuado de ″Persona″ y para un supertipo del tipo más específico del sujeto, ″ingresos″ para el tipo ″Persona″ es el método asignable más específico.
   - Para un sujeto cuyo tipo más específico sea ″Empleado″:
     a. Deje que el método asignable inicial más específico sea ″ingresos″ para el tipo ″Persona″.

b. Itere por el conjunto de métodos asignables. Como el método ″ingresos″ para el tipo ″Empleado″ está definido para un subtipo adecuado de ″Persona″ y para un supertipo del tipo más específico del sujeto (Nota: Un tipo es su propio supertipo y subtipo)el método ″ingresos″ para el tipo ″Empleado″ es una opción mejor para el método asignable más específico. Repita este paso con el método ″ingresos″ para el tipo ″Empleado″ como el método asignable más específico.

c. Como no hay ningún otro método en el conjunto de métodos asignables que esté definido para un subtipo adecuado de ″Empleado″ y para un supertipo del tipo más específico del sujeto, el método ″ingresos″ para el tipo ″Empleado″ es el método asignable más específico.

- Para un sujeto cuyo tipo más específico sea ″Director″:
  a. Deje que el método asignable inicial más específico sea ″ingresos″ para el tipo ″Persona″.
  b. Itere por el conjunto de métodos asignables. Como el método ″ingresos″ para el tipo ″Director″ está definido para un subtipo adecuado de ″Persona″ y para un supertipo del tipo más específico del sujeto (Nota: Un tipo es su propio supertipo y subtipo), el método ″ingresos″ para el tipo ″Director″ es una opción mejor para el método asignable más específico. Repita este paso con el método ″ingresos″ para el tipo ″Director″ como el método asignable más específico.
  c. Como no hay ningún otro método en el conjunto de métodos asignables que esté definido para un subtipo adecuado de ″Director″ y para un supertipo del tipo más específico del sujeto, el método ″ingresos″ para el tipo ″Director″ es el método asignable más específico.

4. Invoque el método asignable más específico.

Información relacionada:

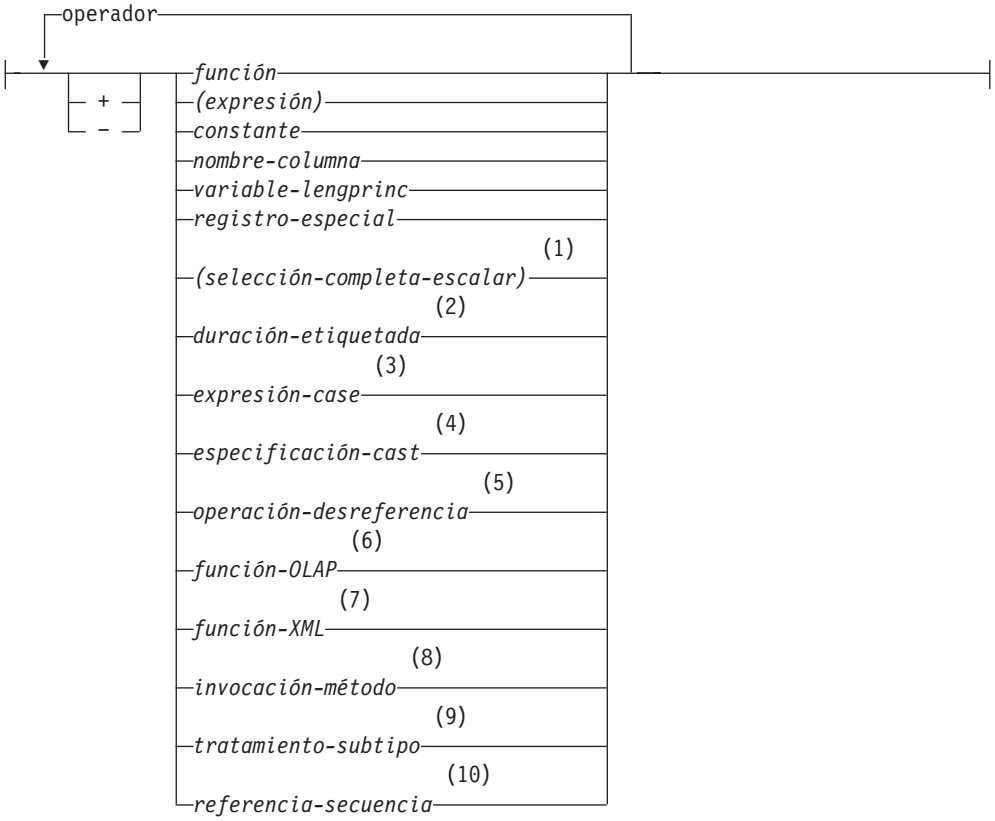
- “Promoción de tipos de datos” en la página 105
- “Asignaciones y comparaciones” en la página 110

Expresiones

Una expresión especifica un valor. Puede ser un valor simple, formado sólo por una constante o un nombre de columna, o puede ser más complejo. Si se utilizan repetidamente expresiones complejas similares, puede plantearse la utilización de una función SQL para encapsular una expresión común.

En una base de datos Unicode, una expresión que acepte una serie de caracteres o gráfica aceptará todo tipo de serie para el que se soporte la conversión.

expresión:



operador:

Notas:

1 Vea “Selección completa escalar” en la página 187 para obtener más información.

2 Vea “Duraciones etiquetadas” en la página 188 para obtener más información.

3 Vea “Expresiones CASE” en la página 193 para obtener más información.

4 Vea “Especificaciones CAST” en la página 195 para obtener más información.

5 Vea “Operaciones de desreferencia” en la página 197 para obtener más información.

6 Vea “Funciones OLAP” en la página 198 para obtener más información.

7 Vea “Funciones XML” en la página 204 para obtener más información.

8 Vea “Invocación de métodos” en la página 212 para obtener más información.

9 Vea “Tratamiento de los subtipos” en la página 214 para obtener más información.

10 Vea “Referencia de secuencia” en la página 214 para obtener más información.

11 || se utiliza como sinónimo de CONCAT.

Expresiones son operadores

Si no se utilizan operadores, el resultado de la expresión es el valor especificado.

Ejemplos:

```
SALARY:SALARY'SALARY'MAX(SALARY)
```

Expresiones con el operador de concatenación

El operador de concatenación (CONCAT) enlaza dos operandos de serie para formar una *expresión de serie*.

Los operandos de la concatenación deben ser series compatibles. Es preciso tener en cuenta que una serie binaria no se puede concatenar con una serie de caracteres, incluso las que se definen como FOR BIT DATA (SQLSTATE 42884).

En una base de datos Unicode, una concatenación que implique operandos de series de caracteres y operandos de series gráficas convertirá primero los operandos de caracteres en operandos gráficos. Observe que, en una base de datos que no sea Unicode, la concatenación no puede implicar operandos de caracteres y operandos gráficos.

Si ambos operandos pueden ser nulos, el resultado también podrá ser nulo; si alguno de ellos es nulo, el resultado es el valor nulo. De lo contrario, el resultado constará de la serie del primer operando seguida por la del segundo. La comprobación se efectúa para detectar los datos mixtos formados defectuosamente al realizar la concatenación.

La longitud del resultado es la suma de las longitudes de los operandos.

El tipo de datos y el atributo de longitud del resultado vienen determinados por los de los operandos, tal como se muestra en la tabla siguiente:

Tabla 12. Tipo de datos y longitud de los operandos concatenados

Operandos	Atributos de longitud combinados	Resultado
CHAR(A) CHAR(B)	<255	CHAR(A+B)
CHAR(A) CHAR(B)	>254	VARCHAR(A+B)
CHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)

Tabla 12. Tipo de datos y longitud de los operandos concatenados (continuación)

Operandos	Atributos de longitud combinados	Resultado
CHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
CHAR(A) LONG VARCHAR	-	LONG VARCHAR
VARCHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)
VARCHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
VARCHAR(A) LONG VARCHAR	-	LONG VARCHAR
LONG VARCHAR LONG VARCHAR	-	LONG VARCHAR
CLOB(A) CHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) VARCHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) LONG VARCHAR	-	CLOB(MIN(A+32K, 2G))
CLOB(A) CLOB(B)	-	CLOB(MIN(A+B, 2G))
GRAPHIC(A) GRAPHIC(B)	<128	GRAPHIC(A+B)
GRAPHIC(A) GRAPHIC(B)	>127	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
GRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC
VARGRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
VARGRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
VARGRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC
LONG VARGRAPHIC LONG VARGRAPHIC	-	LONG VARGRAPHIC
DBCLOB(A) GRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) VARGRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) LONG VARGRAPHIC	-	DBCLOB(MIN(A+16K, 1G))
DBCLOB(A) DBCLOB(B)	-	DBCLOB(MIN(A+B, 1G))
BLOB(A) BLOB(B)	-	BLOB(MIN(A+B, 2G))

Observe que, para que haya compatibilidad con las versiones anteriores, no hay escalada automática de los resultados que implica tipos de datos LONG a los tipos de datos LOB. Por ejemplo, la concatenación de un valor CHAR(200) y un valor LONG VARCHAR totalmente completo da como resultado un error en lugar de una promoción de un tipo de datos CLOB.

La página de códigos del resultado se considera una página de códigos derivada que viene determinada por la página de códigos de sus operandos.

Un operando puede ser un marcador de parámetros. Si se utiliza un marcador de parámetros, el tipo de datos y los atributos de longitud del operando se consideran los mismos que los del operando que no es el marcador de parámetros. El orden de las operaciones tiene su importancia, puesto que determina estos atributos en casos en los que se produce una concatenación anidada.

Ejemplo 1: Si FIRSTNME es Pierre y LASTNAME es Fermat, entonces lo siguiente:

```
FIRSTNME CONCAT ' ' CONCAT LASTNAME
```

devuelve el valor `Pierre Fermat`

Ejemplo 2: Dado:

- `COLA` definido como VARCHAR(5) con valor `'AA'`
- `:host_var` definida como una variable del lenguaje principal con una longitud 5 y el valor `'BB  '`
- `COLC` definido como CHAR(5) con valor `'CC'`
- `COLD` definido como CHAR(5) con valor `'DDDDD'`

El valor de `COLA CONCAT :host_var CONCAT COLC CONCAT COLD` es
`'AABB  CC  DDDDD'`

El tipo de datos es VARCHAR, el atributo de longitud es 17 y la página de códigos resultante es la página de códigos de la base de datos.

Ejemplo 3: Dado:

`COLA` definido como CHAR(10)

`COLB` definido como VARCHAR(5)

El marcador de parámetros de la expresión:

```
COLA CONCAT COLB CONCAT ?
```

se considera VARCHAR(15), porque `COLA CONCAT COLB` se evalúa primero, dando como resultado el primer operando de la segunda operación CONCAT.

Tipos definidos por el usuario

No se puede utilizar un tipo definido por el usuario con el operador de concatenación, aunque sea un tipo diferenciado con un tipo de datos fuente de tipo serie. Para poder concatenar, es preciso crear una función con el operador CONCAT como fuente. Por ejemplo, si existieran los tipos diferenciados TITLE y TITLE_DESCRIPTION y ambos tuvieran los tipos de datos VARCHAR(25), la siguiente función definida por el usuario, ATTACH, se podría utilizar para concatenarlos.

```
CREATE FUNCTION ATTACH (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

También existe la posibilidad de sobrecargar el operador de concatenación empleando una función definida por el usuario para añadir los tipos de datos nuevos.

```
CREATE FUNCTION CONCAT (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

Expresiones con operadores aritméticos

Si se utilizan operadores aritméticos, el resultado de la expresión es un valor derivado de la aplicación de los operadores a los valores de los operandos.

Si cualquier operando puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en sí, el resultado puede ser nulo.

Si algún operando tiene el valor nulo, el resultado de la expresión es el valor nulo.

Los operadores numéricos se pueden aplicar a tipos numéricos con signo y a tipos de fecha y hora (vea "Aritmética de fecha y hora en SQL" en la página 189). Por ejemplo, USER+2 no es válido. Las funciones derivadas se pueden definir para operaciones aritméticas sobre tipos diferenciados con un tipo fuente que sea un tipo numérico con signo.

El operador de prefijo + (más unario) no modifica su operando. El operador de prefijo − (menos unario) invierte el signo de un operando distinto de cero y, si el tipo de datos de A es un entero pequeño, el tipo de datos de −A será un entero grande. El primer carácter del símbolo que sigue a un operador de prefijo no debe ser un signo más ni un signo menos.

Los *operadores infijos* +, −, * y / especifican, respectivamente, una suma, una resta, una multiplicación y una división. El valor del segundo operando de una división no debe ser cero. Estos operadores también se pueden tratar como funciones. Por consiguiente, la expresión *"+"(a,b)* es equivalente a la función de "operador" cuya expresión es *a+b*.

Errores aritméticos

Si se produce un error aritmético como, por ejemplo, una división por cero o se produce un desbordamiento numérico durante el proceso de una expresión, se devuelve un error y la sentencia de SQL que procesa la expresión falla con un error (SQLSTATE 22003 ó 22012).

Una base de datos puede configurarse (utilizando DFT_SQLMATHWARN establecido en sí) para que los errores aritméticos devuelvan un valor nulo para la expresión, emitan un aviso (SQLSTATE 01519 ó 01564) y prosigan con el proceso de la sentencia de SQL. Cuando los errores aritméticos se tratan como nulos, hay implicaciones en los resultados de las sentencias de SQL. A continuación encontrará algunos ejemplos de dichas implicaciones.

- Un error aritmético que se produce en la expresión que es el argumento de una función de columna provoca que se ignore la fila en la determinación del resultado de la función de columna. Si el error aritmético ha sido un desbordamiento, puede afectar de manera significativa a los valores del resultado.
- Un error aritmético que se produce en la expresión de un predicado en una cláusula WHERE puede hacer que no se incluyan filas en el resultado.
- Un error aritmético que se produce en la expresión de un predicado en una restricción de comprobación da como resultado el proceso de actualización o inserción ya que la restricción no es falsa.

Si estos tipos de efectos no son aceptables, deben seguirse pasos adicionales para manejar el error aritmético y producir resultados aceptables. Algunos ejemplos son:

- añadir una expresión case para comprobar la división por cero y establecer el valor deseado para dicha situación

- añadir predicados adicionales para manejar los nulos (por ejemplo, una restricción de comprobación en columnas sin posibilidad de nulos daría:
  ```
  check (c1*c2 is not null and c1*c2>5000)
  ```

  para hacer que la restricción se violase en un desbordamiento).

Dos operandos enteros

Si ambos operandos de un operador aritmético son enteros, la operación se realiza en binario y el resultado es un *entero grande* a no ser que uno de los operandos (o ambos) sea un entero superior, en cuyo caso el resultado es un entero superior. Se pierde cualquier resto de una división. El resultado de una operación aritmética de enteros (incluyendo el menos unitario) debe estar dentro del rango del tipo del resultado.

Operandos enteros y decimales

Si un operando es un entero y el otro es un decimal, la operación se realiza en decimal utilizando una copia temporal del entero que se habrá convertido a número decimal con la precisión *p* y la escala 0; *p* es 19 para un entero superior, 11 para un entero grande y 5 para un entero pequeño.

Dos operandos decimales

Si los dos operandos son decimales, la operación se efectúa en decimal. El resultado de cualquier operación aritmética decimal es un número decimal con una precisión y una escala que dependen de la operación y de la precisión y la escala de los operandos. Si la operación es una suma o una resta y los operandos no tienen la misma escala, la operación se efectúa con una copia temporal de uno de los operandos. La copia del operando más corto se extiende con ceros de cola de manera que la parte de la fracción tenga el mismo número de dígitos que el otro operando.

El resultado de una operación decimal no debe tener una precisión mayor que 31. El resultado de una suma, resta y multiplicación decimal se obtiene de un resultado temporal que puede tener una precisión mayor que 31. Si la precisión del resultado temporal no es mayor que 31, el resultado final es el mismo que el resultado temporal.

Aritmética decimal en SQL

Las fórmulas siguientes definen la precisión y la escala del resultado de las operaciones decimales en SQL. Los símbolos *p* y *s* indican la precisión y la escala del primer operando y los símbolos *p'* y *s'* indican y la precisión y la escala del segundo operando.

Sumas y restas

La precisión es min(31,max(*p*-*s*,*p'*-*s'*) +max(*s*,*s'*)+1). La escala del resultado de una suma o una resta es max (*s*,*s'*).

Multiplicaciones

La precisión del resultado de una multiplicación es min (31,*p*+*p'*) y la escala es min(31,*s*+*s'*).

Divisiones

La precisión del resultado de la división es 31. La escala es *31*-*p*+*s*-*s'*. La escala no debe ser negativa.

Nota: El parámetro de configuración de base de datos MIN_DEC_DIV_3 modifica la escala para las operaciones aritméticas decimales que incluyen la división. Si el valor del parámetro se establece en NO, la escala se calcula como *31-p+s-s*'. Si el parámetro se establece en YES, la escala se calcula como MAX(3, *31-p+ s-s*'). Esto asegura que el resultado de una división decimal tenga siempre una escala de 3 como mínimo (la precisión es siempre 31).

Operandos de coma flotante

Si cualquiera de los dos operandos de un operador aritmético es de coma flotante, la operación se realiza en coma flotante, convirtiendo primero los operandos a números de coma flotante de doble precisión, si es necesario. Por lo tanto, si cualquier elemento de una expresión es un número de coma flotante, el resultado de la expresión es un número de coma flotante de precisión doble.

Una operación en la que intervenga un número de coma flotante y un entero se realiza con una copia temporal del entero que se ha convertido a coma flotante de precisión doble. Una operación en la que intervenga un número de coma flotante y un número decimal se efectúa con una copia temporal del número decimal que se ha convertido a coma flotante de precisión doble. El resultado de una operación de coma flotante debe estar dentro del rango de los números de coma flotante.

Tipos definidos por el usuarios como operandos

Un tipo definido por el usuario no puede utilizarse con operadores aritméticos ni siquiera aunque el tipo de datos fuente sea numérico. Para llevar a cabo una operación aritmética, cree una función con el operador aritmético como fuente. Por ejemplo, si existen los tipos diferenciados INCOME y EXPENSES, y ambos tienen tipos de datos DECIMAL(8,2), se podría utilizar la función REVENUE definida por el usuario para restar uno de otro, de la forma siguiente:

```
CREATE FUNCTION REVENUE (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

El operador - (menos) se puede sobrecargar de forma alternativa utilizando la función definida por el usuario para restar los tipos de datos nuevos.

```
CREATE FUNCTION "-" (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

Selección completa escalar

Una *selección completa escalar*, tal como se utiliza en una expresión, es una selección completa, entre paréntesis, que devuelve una única fila formada por un solo valor de columna. Si la selección completa no devuelve una fila, el resultado de la expresión es el valor nulo. Si el elemento de la lista de selección es una expresión que simplemente es un nombre de columna o una operación de desreferencia, el nombre de columna del resultado está basado en el nombre de la columna.

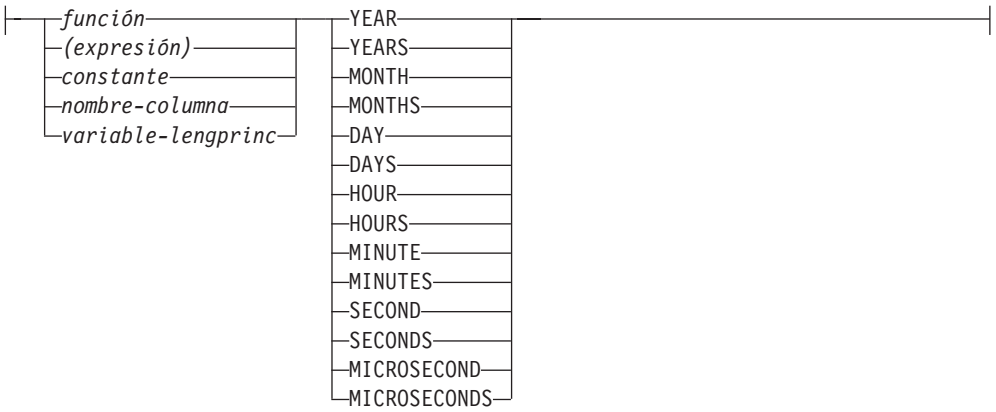
Operaciones de fecha y hora y duraciones

Los valores de fecha y hora se pueden aumentar, disminuir y restar. Estas operaciones pueden incluir números decimales llamados *duraciones*. A continuación se muestra una definición de las duraciones y una especificación de las reglas para la aritmética de la fecha y hora.

Una duración es un número que representa un intervalo de tiempo. Existen cuatro tipos de duraciones:

Duraciones etiquetadas

duración-etiquetada:



Una *duración etiquetada* representa una unidad de tiempo específica expresada por un número (que puede ser el resultado de una expresión) seguido de una de las siete palabras clave de duración: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS o MICROSECONDS. (También se acepta la forma singular de estas palabras clave: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND y MICROSECOND.) El número especificado se convierte como si se asignara a un número DECIMAL(15,0). Sólo puede utilizarse una duración etiquetada como operando de un operador aritmético en el que el otro operando sea un valor de tipo de datos DATE, TIME o TIMESTAMP. Así pues, la expresión HIREDATE + 2 MONTHS + 14 DAYS es válida, mientras que la expresión HIREDATE + (2 MONTHS + 14 DAYS) no lo es. En ambas expresiones, las duraciones etiquetadas son 2 MONTHS (meses) y 14 DAYS (días).

Duración de fecha

Una *duración de fecha* representa un número de años, meses y días, expresados como un número DECIMAL(8,0). Para que se interprete correctamente, el número debe tener el formato *aaaammdd.*, donde *aaaa* representa el número de años, *mm* el número de meses y *dd* el número de días. (El punto en el formato indica un tipo de datos DECIMAL.) El resultado de restar un valor de fecha de otro, como sucede en la expresión HIREDATE – BRTHDATE, es una duración de fecha.

Duración de hora

Una *duración de hora* representa un número de horas, minutos y segundos, expresado como un número DECIMAL(6,0). Para interpretarse correctamente, el número debe tener el formato *hhmmss.*, donde *hh* representa el número de horas, *mm* el número de minutos y *ss* el número de segundos. (El punto en el formato indica un tipo de datos DECIMAL.) El resultado de restar un valor de hora de otro es una duración de hora.

Duración de la indicación de fecha y hora

Una *duración de la indicación de fecha y hora* representa un número de años, meses, días, horas, minutos, segundos y microsegundos expresado como un número DECIMAL(20,6). Para que se interprete correctamente, el número debe tener el formato *aaaammddhhmmss.zzzzzz*, donde *aaaa*, *mm*, *dd*, *hh*, *mm*, *ss* y *zzzzzz* representan el número de años, meses, días, horas, minutos, segundos y microsegundos respectivamente. El resultado de restar un valor de indicación de fecha y hora de otro es una duración de la indicación de fecha y hora.

Aritmética de fecha y hora en SQL

Las únicas operaciones aritméticas que pueden efectuarse con valores de fecha y hora son la suma y la resta. Si un valor de fecha y hora es un operando de suma, el otro operando debe ser una duración. A continuación, encontrará las reglas específicas que rigen la utilización del operador de suma con valores de fecha y hora.

- Si un operando es una fecha, el otro operando debe ser una duración de fecha o una duración etiquetada de YEARS, MONTHS o DAYS.
- Si un operando es una hora, el otro operando debe ser una duración de hora o una duración etiquetada de HOURS, MINUTES o SECONDS.
- Si un operando es una fecha y hora, el otro operando debe ser una duración. Cualquier tipo de duración es válido.
- Ningún operando del operador de suma puede ser un marcador de parámetros.

Las normas para la utilización del operador de resta con valores de fecha y hora no son las mismas que para la suma, porque un valor de fecha y hora no puede restarse de una duración, y porque la operación de restar dos valores de fecha y hora no es la misma que la operación de restar una duración de un valor de fecha y hora. A continuación se muestran las normas específicas que rigen la utilización del operador de resta con valores de fecha y hora.

- El primer operando es una fecha, el segundo operando debe ser una fecha, una duración de fecha, una representación de una fecha en forma de serie o una duración etiquetada de YEARS, MONTHS o DAYS.
- Si el segundo operando es una fecha, el primer operando debe ser una fecha o una representación de una fecha en forma de serie.
- Si el primer operando es una hora, el segundo operando debe ser una hora, una duración de hora, una representación de una hora en forma de serie o una duración etiquetada de HOURS, MINUTES o SECONDS.
- Si el segundo operando es una hora, el primer operando debe ser una hora o una representación de una hora en forma de serie.
- Si el primer operando es una fecha y hora, el segundo operando debe ser una fecha y hora, una representación de una fecha y hora en forma de serie o una duración.
- Si el segundo operando es una fecha y hora, el primer operando debe ser una fecha y hora o una representación de una fecha y hora en forma de serie.
- Ningún operando del operador de resta puede ser un marcador de parámetros.

Aritmética de fecha

Las fechas se pueden restar, aumentar o disminuir.

Resta de fechas: Al restar un fecha (DATE2) de otra (DATE1) se obtiene como resultado una duración de fecha que especifica el número de años, meses y días entre las dos fechas. El tipo de datos del resultado es DECIMAL(8,0). Si DATE1 es mayor o igual que DATE2, DATE2 se resta de DATE1. Si DATE1 es menor que DATE2, DATE1 se resta de DATE2 y el signo del resultado se convierte en negativo. La descripción siguiente clarifica los pasos que intervienen en el resultado de la operación = DATE1 − DATE2.

```
   Si DAY(DATE2) <= DAY(DATE1)
   entonces DAY(RESULT) = DAY(DATE1) - DAY(DATE2).
Si DAY(DATE2) > DAY(DATE1)     entonces DAY(RESULT) = N + DAY(DATE1)
- DAY(DATE2)
  donde N = el último día de MONTH(DATE2).
  MONTH(DATE2) se aumenta en 1.
```

```
Si MONTH(DATE2) <= MONTH(DATE1)     entonces MONTH(RESULT) = MONTH(DATE1)
- MONTH(DATE2).
Si MONTH(DATE2) > MONTH(DATE1)     entonces MONTH(RESULT) = 12 + MONTH(DATE1)
- MONTH(DATE2).
   YEAR(DATE2) se aumenta en 1.
   YEAR(RESULT) = YEAR(DATE1) - YEAR(DATE2).
```

Por ejemplo, el resultado de DATE('15/3/2000') – '31/12/1999' es 00000215. (o una duración de 0 años, 2 meses y 15 días).

Incremento y disminución de las fechas: Al añadir o restar una duración a una fecha se obtiene como resultado también una fecha. (En esta operación, un mes equivale a una página de un calendario. La adición de meses a una fecha es como ir pasando páginas a un calendario, empezando por la página en la que aparece la fecha.) El resultado debe estar comprendido entre las fechas 1 de enero de 0001 y 31 de diciembre de 9999, ambos inclusive.

Si se suma o resta una duración de años, solamente la parte de la fecha correspondiente a los años se verá afectada. Tanto el mes como el día permanecen inalterados, a no ser que el resultado fuera el 29 de febrero en un año no bisiesto. En este caso, el día se cambia a 28 y se define un indicador de aviso en la SQLCA para indicar el ajuste.

Del mismo modo, si se suma o resta una duración de meses, solamente los meses, y los años si fuera necesario, se verán afectados. La parte de una fecha correspondiente a los años no se cambia a no ser que el resultado no fuera válido (31 de setiembre, por ejemplo). En este caso, el día se establece en el último día del mes y se define un indicador de aviso en la SQLCA para indicar el ajuste.

Al añadir o restar una duración de días afectará, obviamente, a la parte de la fecha correspondiente a los días y potencialmente al mes y al año.

Las duraciones de fecha, ya sean positivas o negativas, también se pueden añadir y restar a las fechas. Tal como ocurre con las duraciones etiquetadas, se obtiene como resultado una fecha válida y se define un indicador de aviso en la SQLCA siempre que se deba efectuar un ajuste de fin de mes.

Cuando se suma una duración de fecha positiva a una fecha, o una duración de fecha negativa se resta de una fecha, la fecha aumenta el número especificado de años, meses y días, en ese orden. Así pues, DATE1 + X, donde X es un número DECIMAL(8,0) positivo, equivale a la expresión:

```
DATE1 + YEAR(X) YEARS + MONTH(X) MONTHS + DAY(X) DAYS.
```

Cuando una duración de fecha positiva se resta de una fecha, o bien se añade una duración de fecha negativa a una fecha, la fecha disminuye en el número días, meses y años especificados, en este orden. Así pues, DATE1 – X, donde X es un número DECIMAL(8,0) positivo, equivale a la expresión:

```
DATE1 - DAY(X) DAYS - MONTH(X) MONTHS - YEAR(X) YEARS.
```

Al añadir duraciones a fechas, la adición de un mes a una fecha determinada da la misma fecha un mes posterior a menos que la fecha no exista en el siguiente mes. En este caso, se establece la fecha correspondiente al último día del siguiente mes. Por ejemplo, 28 de enero más un mes da como resultado 28 de febrero y si se añade un mes al 29, 30 ó 31 de enero también se obtendrá como resultado el 28 de febrero o bien 29 de febrero si se trata de un año bisiesto.

Nota: Si se añade uno o más meses a una fecha determinada y del resultado se resta la misma cantidad de meses, la fecha final no tiene por qué ser necesariamente la misma que la original.

Aritmética de las horas

Las horas se pueden restar, aumentar o disminuir.

Resta de los valores de horas: El resultado de restar una hora (HOUR2) de otra (HOUR1) es una duración que especifica el número de horas, minutos y segundos entre las dos horas. El tipo de datos del resultado es DECIMAL(6,0).

Si HOUR1 es mayor o igual que HOUR2, HOUR2 se resta de HOUR1.

Si HOUR1 es menor que HOUR2, HOUR1 se resta de HOUR2 y el signo del resultado se convierte en negativo. La descripción siguiente clarifica los pasos que intervienen en el resultado de la operación = HOUR1 − HOUR2.

```
Si SECOND(TIME2) <= SECOND(TIME1)
entonces SECOND(RESULT) = SECOND(HOUR1) - SECOND(HOUR2).
Si SECOND(TIME2) > SECOND(TIME1) entonces SECOND(RESULT) =
60 + SECOND(HOUR1) - SECOND(HOUR2).
MINUTE(HOUR2) se aumenta entonces en 1.
Si MINUTE(TIME2) <= MINUTE(TIME1)entonces MINUTE(RESULT) = MINUTE(HOUR1)
- MINUTE(HOUR2).
Si MINUTE(TIME1) > MINUTE(TIME1) entonces MINUTE(RESULT) =
60 + MINUTE(HOUR1) - MINUTE(HOUR2).
HOUR(HOUR2) se aumenta entonces en 1.
HOUR(RESULT) = HOUR(TIME1) - HOUR(TIME2).
```

Por ejemplo, el resultado de TIME('11:02:26') − '00:32:56' es 102930. (una duración de 10 horas, 29 minutos y 30 segundos).

Incremento y disminución de los valores de horas: El resultado de sumar una duración a una hora, o de restar una duración de una hora, es una hora. Se rechaza cualquier desbordamiento o subdesbordamiento de horas, garantizando de este modo que el resultado sea siempre una hora. Si se suma o resta una duración de horas, sólo se ve afectada la parte correspondiente a las horas. Los minutos y los segundos no cambian.

De manera parecida, si se suma o resta una duración de minutos, sólo se afecta a los minutos y, si fuera necesario, a las horas. La parte correspondiente a los segundos no cambia.

Al añadir o restar una duración de segundos afectará, obviamente, a la parte de la fecha correspondiente a los segundos y potencialmente a los minutos y a las horas.

Las duraciones de hora, tanto positivas como negativas, pueden también sumarse y restarse a las horas. El resultado es una hora que se ha incrementado o disminuido en el número especificado de horas, minutos y segundos, por ese orden. `TIME1 + X`, donde "X" es un número DECIMAL(6,0), es equivalente a la expresión:

```
TIME1 + HOUR(X) HOURS + MINUTE(X) MINUTES + SECOND(X) SECONDS
```

Nota: Aunque la hora '24:00:00' se acepta como una hora válida, no se devuelve nunca como resultado de una suma o resta de horas, ni siquiera aunque el operando de duración sea cero (por ejemplo, hora('24:00:00')±0 segundos = '00:00:00').

Aritmética de la indicación de fecha y hora

Las indicaciones de fecha y hora se pueden restar, incrementar o disminuir.

Resta de indicaciones de fecha y hora: El resultado de restar una indicación de fecha y hora (TS2) de otra (TS1) es una duración de la indicación de fecha y hora que especifica el número de años, meses, días, horas, minutos, segundos y microsegundos entre las dos indicaciones de fecha y hora. El tipo de datos del resultado es DECIMAL(20,6).

Si TS1 es mayor o igual que TS2, TS2 se resta de TS1. Si TS1 es menor que TS2, TS1 se resta de TS2 y el signo del resultado se convierte en negativo. La descripción siguiente clarifica los pasos que intervienen en el resultado de la operación = TS1 − TS2:

```
Si MICROSECOND(TS2) <= MICROSECOND(TS1)
entonces MICROSECOND(RESULT) = MICROSECOND(TS1) -
MICROSECOND(TS2).
Si MICROSECOND(TS2) > MICROSECOND(TS1) entonces MICROSECOND(RESULT) =
1000000 + MICROSECOND(TS1) - MICROSECOND(TS2)
y SECOND(TS2) se aumenta en 1.
```

La parte correspondiente a los segundos y minutos de la indicación de fecha y hora se resta tal como se especifica en las reglas para la resta de horas.

```
Si HOUR(TS2) <= HOUR(TS1)
entonces HOUR(RESULT) = HOUR(TS1) - HOUR(TS2).
Si HOUR(TS2) > HOUR(TS1) entonces HOUR(RESULT) = 24 + HOUR(TS1) - HOUR(TS2)
y DAY(TS2) se aumenta en 1.
```

La parte correspondiente a la fecha de las indicaciones de fecha y hora se resta tal como se especifica en las reglas para la resta de fechas.

Incremento y disminución de indicaciones de fecha y hora: El resultado de sumar o restar una duración con una indicación de fecha y hora es también una indicación de fecha y hora. El cálculo con fechas y horas se realiza tal como se ha definido anteriormente, excepto que se acarrea un desbordamiento o subdesbordamiento a la parte de fecha del resultado, que debe estar dentro del rango de fechas válidas. El desbordamiento de microsegundos pasa a segundos.

Prioridad de las operaciones

Las expresiones entre paréntesis y las expresiones de desreferencia se evalúan primero de izquierda a derecha. (Los paréntesis también se utilizan en sentencias de subselección, condiciones de búsqueda y funciones. Sin embargo, no deben utilizarse para agrupar arbitrariamente secciones dentro de sentencias de SQL.) Cuando del orden de evaluación no se especifica mediante paréntesis, los operadores de prefijo se aplican antes que la multiplicación y división, y la multiplicación y división se aplican antes que la suma y la resta. Los operadores de un mismo nivel de prioridad se aplican de izquierda a derecha.

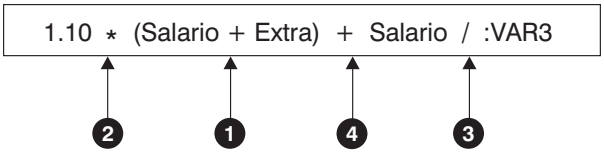
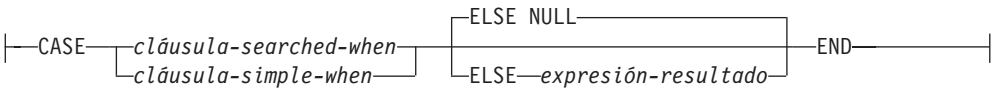


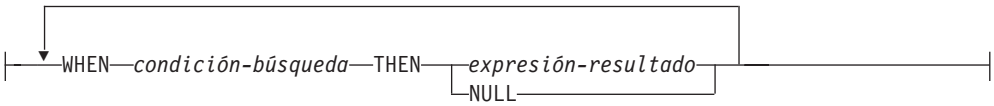
Figura 11. Prioridad de las operaciones

Expresiones CASE

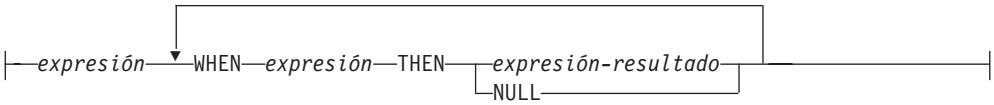
expresión-case:



cláusula-searched-when:



cláusula-simple-when:



Las expresiones CASE permiten seleccionar una expresión en función de la evaluación de una o varias condiciones. En general, el valor de la expresión-case es el valor de la *expresión-resultado* que sigue a la primera (más a la izquierda) expresión case que se evalúa como cierta. Si ninguna se evalúa como cierta y está presente la palabra clave ELSE, el resultado es el valor de la *expresión-resultado* o NULL. Si ninguna se evalúa como cierta y no se utiliza la palabra clave ELSE, el resultado es NULL. Tenga presente que cuando una expresión CASE se evalúa como desconocida (debido a valores NULL), la expresión CASE no es cierta y por eso se trata igual que una expresión CASE que se evalúa como falsa.

Si la expresión CASE está en una cláusula VALUES, un predicado IN, una cláusula GROUP BY o en una cláusula ORDER BY, la *condición-búsqueda* de una cláusula-searched-when no puede ser un predicado cuantificado, un predicado IN que hace uso de una selección completa ni un predicado EXISTS (SQLSTATE 42625).

Cuando se utiliza la *cláusula-simple-when*, se comprueba si el valor de la *expresión* anterior a la primera palabra clave *WHEN* es igual al valor de la *expresión* posterior a la palabra clave WHEN. Por lo tanto, el tipo de datos de la *expresión* anterior a la primera palabra clave *WHEN* debe ser comparable a los tipos de datos de cada *expresión* posterior a la palabra o palabras clave WHEN. La *expresión* anterior a la primera palabra clave *WHEN* de una *cláusula-simple-when* no puede incluir ninguna función que sea una variante o que tenga una acción externa (SQLSTATE 42845).

Una *expresión-resultado* es una *expresión* que sigue a las palabras clave THEN o ELSE. Debe haber, como mínimo, una *expresión-resultado* en la expresión CASE (NULL no puede especificarse para cada case) (SQLSTATE 42625). Todas las expresiones-resultado deben tener tipos de datos compatibles (SQLSTATE 42804).

Ejemplos:

- Si el primer carácter de un número de departamento corresponde a una división dentro de la organización, se puede utilizar una expresión CASE para listar el nombre completo de la división a la que pertenece cada empleado:

```
SELECT EMPNO, LASTNAME,
     CASE SUBSTR(WORKDEPT,1,1)
     WHEN 'A' THEN 'Administración'
     WHEN 'B' THEN 'Recursos humanos'
     WHEN 'C' THEN 'Contabilidad'
     WHEN 'D' THEN 'Diseño'
     WHEN 'E' THEN 'Operaciones'
   END
   FROM EMPLOYEE;
```

- El número de años de formación académica se usa en la tabla EMPLOYEE para obtener el nivel de formación. Una expresión CASE se puede utilizar para agrupar estos datos y para mostrar el nivel de formación.

```
SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME,
     CASE
     WHEN EDLEVEL < 15 THEN 'SECONDARY'
     WHEN EDLEVEL < 19 THEN 'COLLEGE'
     ELSE 'POST GRADUATE'
   END
   FROM EMPLOYEE
```

- Otro ejemplo interesante del uso de una expresión CASE consiste en la protección de los errores que surjan de una división por 0. Por ejemplo, el siguiente código detecta los empleados que perciben más de un 25% de sus ingresos en comisiones, pero que su sueldo no se basa enteramente en comisiones.

```
SELECT EMPNO, WORKDEPT, SALARY+COMM FROM EMPLOYEE
WHERE (CASE WHEN SALARY=0 THEN NULL
  ELSE COMM/SALARY
  END) > 0.25;
```

- Las siguientes expresiones CASE son iguales:

```
SELECT LASTNAME,
  CASE
  WHEN LASTNAME = 'Haas' THEN 'Presidente'
  ...

SELECT LASTNAME,
  CASE LASTNAME
  WHEN 'Haas' THEN 'Presidente'
  ...
```

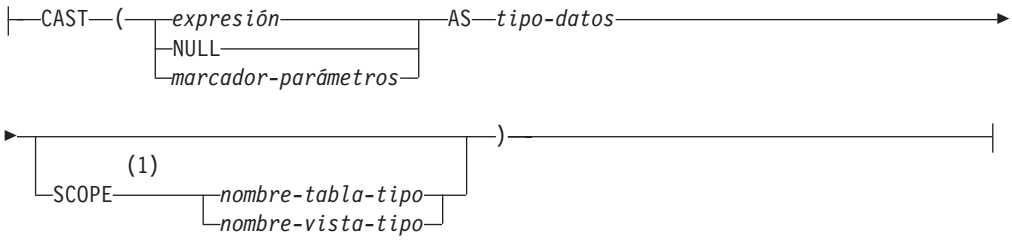
Existen dos funciones escalares, NULLIF y COALESCE, que sirven exclusivamente para manejar un subconjunto de la funcionalidad que una expresión CASE puede ofrecer. La Tabla 13 muestra la expresión equivalente al utilizar CASE o estas funciones.

Tabla 13. Expresiones CASE equivalentes

Expresión	Expresión equivalente
CASE WHEN e1=e2 THEN NULL ELSE e1 END	NULLIF(e1,e2)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE e2 END	COALESCE(e1,e2)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE COALESCE(e2,...,eN) END	COALESCE(e1,e2,...,eN)

Especificaciones CAST

especificación-cast:



Notas:

1 La cláusula SCOPE sólo se aplica al tipo de datos REF.

La especificación CAST devuelve el operando cast (el primer operando) convertido al tipo especificado por el *tipo de datos*. Si no se soporta cast, se devuelve un error (SQLSTATE 42846).

expresión
Si el operando cast es una expresión (distinta del marcador de parámetros o NULL), el resultado es el valor del argumento convertido al *tipo de datos* de destino especificado.

Al convertir series de caracteres (que no sean CLOB) en una serie de caracteres de longitud diferente, se devuelve un aviso (SQLSTATE 01004) si se truncan otros caracteres que no sean los blancos de cola. Al convertir series de caracteres gráficas (que no sean DBCLOB) en una serie de caracteres gráfica con una longitud diferente, se devuelve un aviso (SQLSTATE 01004) si se truncan otros caracteres que no sean los blancos de cola. Para los operandos BLOB, CLOB y DBCLOB de cast, el mensaje de aviso aparece si se trunca cualquier carácter.

NULL
Si el operando cast es la palabra clave NULL, el resultado es un valor nulo que tiene el *tipo de datos* especificado.

marcador-parámetros
Un marcador de parámetros (especificado como un signo de interrogación) se suele considerar como una expresión pero se documenta independientemente en este caso porque tiene un significado especial. Si el operando cast es un *marcador-parámetros*, el *tipo de datos* especificado se considera una promesa de que la sustitución se podrá asignar al tipo de datos especificado (utilizando la asignación de almacenamiento para series). Un marcador de parámetros como este se considera un *marcador de parámetros con tipo*. Los marcadores de parámetros con tipo se tratan como cualquier otro valor con tipo en lo referente a la resolución de la función, a DESCRIBE de una lista de selección o a la asignación de columnas.

tipo de datos
Nombre de un tipo de datos existente. Si el nombre de tipo no está calificado, la vía de acceso de SQL se utiliza para realizar la resolución del tipo de datos. Un tipo de datos que tenga asociados atributos como, por ejemplo, la longitud o la precisión y escala debe incluir dichos atributos al especificar el *tipo de datos* (CHAR toma por omisión la longitud de 1 y DECIMAL toma por omisión una precisión de 5 y una escala de 0 si no se especifican). Las restricciones sobre los tipos de datos soportados se basan en el operando cast especificado.

- Para un operando cast que sea una *expresión*, los tipos de datos de destino a los que se da soporte dependen del tipo de datos del operando cast (tipo de datos fuente).
- Para un operando cast que sea la palabra clave NULL se puede utilizar cualquier tipo de datos existente.
- Para un operando cast que sea un marcador de parámetros, el tipo de datos de destino puede ser cualquier tipo de datos existente. Si el tipo de datos es un tipo diferenciado definido por el usuario, la aplicación que hace uso del marcador de parámetros utilizará el tipo de datos fuente del tipo diferenciado definido por el usuario. Si el tipo de datos es un tipo estructurado definido por el usuario, la aplicación que hace uso del marcador de parámetros utilizará el tipo de parámetro de entrada de la función de transformación TO de SQL para el tipo estructurado definido por el usuario.

SCOPE
Cuando el tipo de datos es un tipo de referencia, puede definirse un ámbito que identifique la tabla de destino o la vista de destino de la referencia.

nombre-tabla-tipo
El nombre de una tabla con tipo. Ya debe existir la tabla (SQLSTATE 42704). La conversión debe hacerse hacia el *tipo-datos* REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM).

nombre-vista-tipo
El nombre de una vista con tipo. La vista debe existir o tener el mismo nombre que la vista a crear que incluye la conversión del tipo de datos como parte de la definición de la vista (SQLSTATE 42704). La conversión debe hacerse hacia el *tipo-datos* REF(*S*), donde *S* es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM).

Cuando se convierten datos numéricos en datos de caracteres, el tipo de datos resultante es una serie de caracteres de longitud fija. . Cuando se convierten datos de caracteres en datos numéricos, el tipo de datos resultante depende del tipo de número especificado. Por ejemplo, si se convierte hacia un entero, pasará a ser un entero grande. .

Ejemplos:

- A una aplicación sólo le interesa la parte entera de SALARY (definido como decimal (9,2)) de la tabla EMPLOYEE. Se podría preparar la siguiente consulta, con el número de empleado y el valor del entero de SALARY.

```
SELECT EMPNO, CAST(SALARY AS INTEGER) FROM EMPLOYEE
```

- Supongamos que hay un tipo diferenciado denominado T_AGE que se define como SMALLINT y se utiliza para crear la columna AGE en la tabla PERSONNEL. Supongamos también que existe también un tipo diferenciado denominado R_YEAR que está definido en INTEGER y que se utiliza para crear la columna RETIRE_YEAR en la tabla PERSONNEL. Se podría preparar la siguiente sentencia de actualización.

```
UPDATE PERSONNEL SET RETIRE_YEAR =?
  WHERE AGE = CAST( ? AS T_AGE)
```

  El primer parámetro es un marcador de parámetros no tipificado que tendría un tipo de datos de R_YEAR, si bien la aplicación utilizará un entero para este marcador de parámetros. Esto no necesita la especificación explícita de CAST porque se trata de una asignación.

  El segundo marcador de parámetros es un marcador de parámetros con tipo que se convierte como un tipo diferenciado T_AGE. Esto cumple el requisito de que

la comparación debe realizarse con tipos de datos compatibles. La aplicación utilizará el tipo de datos fuente (que es SMALLINT) para procesarlo con este marcador de parámetros.

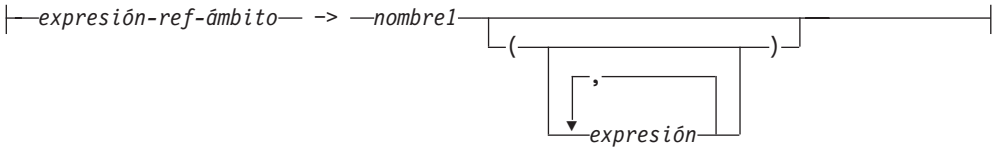
El proceso satisfactorio de esta sentencia supone que la vía de acceso de función incluye el nombre de esquema del esquema (o esquemas) donde están definidos los dos tipos diferenciados.

- Una aplicación suministra un valor que es una serie de bits como, por ejemplo, una corriente de audio y no debería realizarse la conversión de página de códigos antes de que se utilice en una sentencia SQL. La aplicación podría utilizar la siguiente función CAST:

```
CAST( ? AS VARCHAR(10000) FOR BIT DATA)
```

Operaciones de desreferencia

operación-desreferencia:



El ámbito de la expresión de referencia con ámbito es una tabla o vista llamada tabla o vista *destino*. La expresión de referencia con ámbito identifica una *fila destino*. La *fila destino* es la fila de la tabla o vista destino (o de una sus subtablas o subvistas) cuyo valor de la columna de identificador de objeto (OID) coincide con la expresión de referencia. Se puede utilizar la operación de desreferencia para acceder a una columna de la fila destino, o para invocar un método, utilizando la fila destino como sujeto del método. El resultado de una operación de desreferencia puede siempre ser nulo. La operación de desreferencia tiene prioridad por encima de todos los otros operadores.

expresión-ref-ámbito
Una expresión que es un tipo de referencia que tiene un ámbito (SQLSTATE 428DT). Si la expresión es una variable del lenguaje principal, un marcador de parámetros u otro valor de tipo de referencia sin ámbito, se necesita una especificación CAST con una cláusula SCOPE para proporcionar un ámbito a la referencia.

nombre1
Especifica un identificador no calificado.

Si *nombre1* no va seguido por ningún paréntesis y *nombre1* coincide con el nombre de un atributo del tipo destino, el valor de la operación de desreferencia es el valor de la columna mencionada de la fila destino. En este caso, el tipo de datos de la columna (que puede contener nulos) determina el tipo del resultado de la operación de desreferencia. Si no existe ninguna fila destino cuyo identificador de objeto coincida con la expresión de referencia, el resultado de la operación de desreferencia es nulo. Si la operación de desreferencia se utiliza en una lista de selección y no se incluye como parte de una expresión, *nombre1* pasa a ser el nombre de la columna resultante.

Si *nombre1* va seguido por un paréntesis o *nombre1* no coincide con el nombre de un atributo del tipo destino, la operación de desreferencia se trata como una invocación de método. El nombre del método invocado es *nombre1*. El sujeto del método es la fila destino, que se considera como una instancia de su tipo

estructurado. Si no existe ninguna fila destino cuyo identificador de objeto coincida con la expresión de referencia, el sujeto del método es un valor nulo del tipo destino. Las expresiones entre paréntesis, si las hay, proporcionan los restantes parámetros de la invocación del método. El proceso normal se utiliza para la resolución de la invocación del método. El tipo resultante del método seleccionado (que puede contener nulos) determina el tipo resultante de la operación de desreferencia.

El ID de autorización de la sentencia que utiliza una operación de desreferencia debe tener el privilegio SELECT sobre la tabla de destino de la *expresión-ref-ámbito* (SQLSTATE 42501).

Una operación de desreferencia no puede nunca modificar valores de la base de datos. Si se utiliza una operación de desreferencia para invocar un método mutador, éste modifica una copia de la fila destino y devuelve la copia, dejando inalterada la base de datos.

Ejemplos:

- Suponga que existe una tabla EMPLOYEE que contiene una columna denominada DEPTREF, que es un tipo de referencia con ámbito para una tabla con tipo basada en un tipo que incluye el atributo DEPTNAME. Los valores de DEPTREF de la tabla EMPLOYEE deben corresponderse con los valores de la columna de OID de la tabla de destino de la columna DEPTREF.

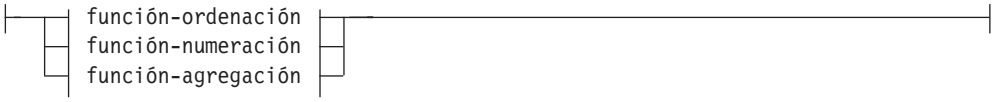
```
SELECT EMPNO, DEPTREF->DEPTNAME
  FROM EMPLOYEE
```

- Utilizando las mismas tablas que en el ejemplo anterior, utilice una operación de desreferencia para invocar un método llamado BUDGET, con la fila destino como parámetro sujeto y '1997' como parámetro adicional.

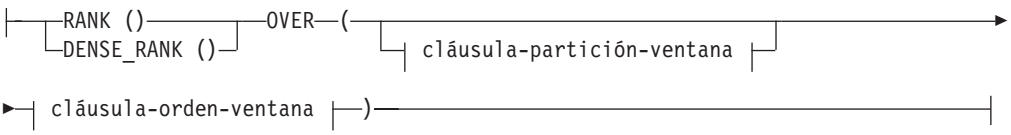
```
SELECT EMPNO, DEPTREF->BUDGET('1997') AS DEPTBUDGET97
  FROM EMPLOYEE
```

Funciones OLAP

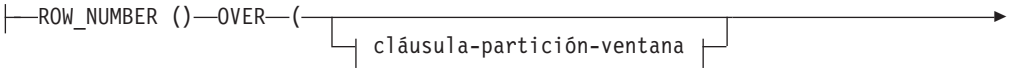
función-OLAP:



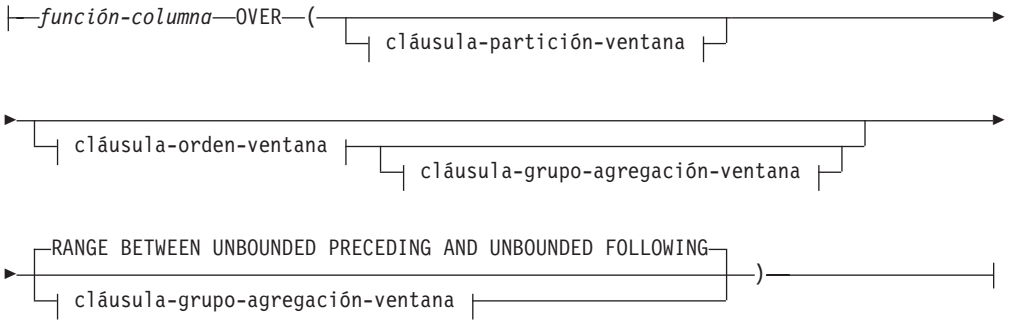
función-ordenación:



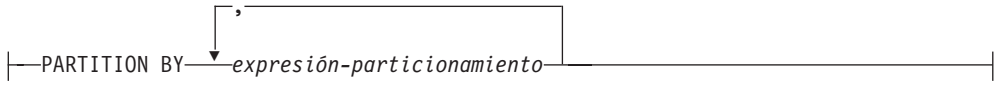
función-numeración:



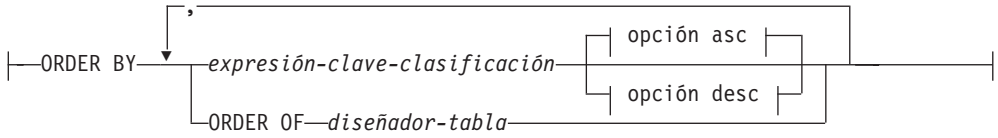
función-agregación:



cláusula-partición-ventana:



cláusula-orden-ventana:



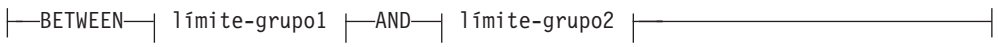
opción asc:

opción desc:

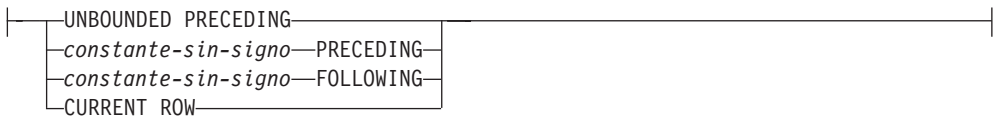
cláusula-grupo-agregación-ventana:

inicio-grupo:

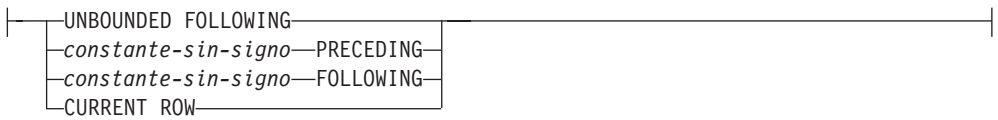
entre-grupo:



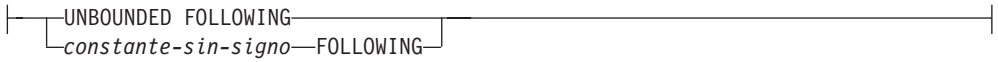
límite-grupo1:



límite-grupo2:



final-grupo:



Las funciones OLAP (On-Line Analytical Processing) devuelven información sobre ordenación y numeración de filas y sobre funciones de columna existentes, así como un valor escalar en el resultado de una consulta. Se puede incluir una función OLAP en expresiones, en una lista de selección o en la cláusula ORDER BY de una sentencia SELECT (SQLSTATE 42903). Las funciones OLAP no se pueden utilizar como argumento de una función de columna (SQLSTATE 42607). La función OLAP se aplica a la tabla resultante de la subselección más interna donde reside la función OLAP.

Cuando se utiliza una función OLAP, se especifica una ventana que define las filas a las que se aplica la función, y en qué orden. Cuando la función OLAP se utiliza con una función de columna, las filas pertinentes se pueden definir con más detalle, con respecto a la fila actual, en forma de rango o indicando un número de filas que preceden y siguen a la fila actual. Por ejemplo, dentro de una división por meses, se puede calcular un valor promedio respecto a los tres meses anteriores.

La función de ordenación calcula la posición ordinal de una fila dentro de la ventana. Las filas que no son distintas con respecto a la ordenación dentro de sus ventanas tienen asignada la misma posición. Los resultados de la ordenación se pueden definir con o sin huecos en los números que resultan de valores duplicados.

Si se especifica RANK, la posición de una fila se define como 1 más el número de filas que preceden estrictamente a la fila. Por lo tanto, si dos o más filas no difieren con respecto a la ordenación, habrá uno o más huecos en la numeración jerárquica secuencial.

Si se especifica DENSE_RANK (o DENSERANK), el rango de una fila se define como 1 más el número de filas que la preceden que son distintas respecto a la ordenación. Por tanto, no habrá huecos en la numeración jerárquica secuencial.

La función ROW_NUMBER (o ROWNUMBER) calcula el número secuencial de la fila dentro de la ventana definida por la ordenación, empezando por 1 para la primera fila. Si la cláusula ORDER BY no está especificada en la ventana, los números de fila se asignan a las filas en un orden arbitrario, tal como son devueltas por la subselección (no de acuerdo con ninguna cláusula ORDER BY de la sentencia-select).

El tipo de datos del resultado de RANK, DENSE_RANK o ROW_NUMBER es BIGINT. El resultado no puede ser nulo.

PARTITION BY (*expresión-particionamiento***,...)**
Define la partición que se utiliza para aplicar la función. Una *expresión-particionamiento* es una expresión utilizada para definir el particionamiento del conjunto resultante. Cada *nombre-columna* referenciado en una expresión-particionamiento debe identificar, sin ambigüedades, una columna del conjunto resultante de la sentencia de subselección donde reside la función OLAP (SQLSTATE 42702 ó 42703). Una expresión-particionamiento no puede incluir una selección completa escalar (SQLSTATE 42822) ni ninguna función que no sea determinista o que tenga una acción externa (SQLSTATE 42845).

ORDER BY (*expresión-clave-clasificación***,...)**
Define la ordenación de las filas dentro de una partición que determina el valor de la función OLAP o el significado de los valores de fila en la cláusula-grupo-agregación-ventana (no define la ordenación del conjunto resultante de la consulta).

expresión-clave-clasificación
Una expresión utilizada para definir la ordenación de las filas dentro de una partición de ventana. Cada nombre de columna referenciado en una expresión-clave-clasificación debe identificar, sin ambigüedades, una columna del conjunto resultante de la subselección, incluida la función OLAP (SQLSTATE 42702 ó 42703). Una expresión-clave-clasificación no puede incluir una selección completa escalar (SQLSTATE 42822) ni una función que no sea determinista o que tenga una acción externa (SQLSTATE 42845). Esta cláusula es necesaria para las funciones RANK y DENSE_RANK (SQLSTATE 42601).

ASC
Utiliza los valores de la expresión-clave-clasificación en orden ascendente.

DESC
Utiliza los valores de la expresión-clave-clasificación en orden descendente.

NULLS FIRST
La ordenación de la ventana tiene en cuenta los valores nulos *antes* de todos los valores no nulos en el orden de clasificación.

NULLS LAST
La ordenación de la ventana tiene en cuenta los valores nulos *después* de todos los valores no nulos en el orden de clasificación.

ORDER OF *diseñador-tabla*
Especifica que debe aplicarse el mismo orden utilizado en *diseñador-tabla* a la tabla resultante de la subselección. Debe haber una referencia de tabla que se corresponda con *diseñador-tabla* en la cláusula FROM de la subselección que especifica esta cláusula (SQLSTATE 42703). La subselección (o selección completa) correspondiente al *diseñador-tabla* especificado debe incluir una cláusula ORDER BY que dependa de los datos (SQLSTATE 428FI). El orden que se aplica es el mismo que si las columnas de la cláusula ORDER BY de la subselección anidada (o selección completa) se incluyeran en la subselección exterior (o selección completa) y estas columnas se especificaran en lugar de la cláusula ORDER OF.

cláusula-grupo-agregación-ventana
El grupo de agregación de una fila R es un conjunto de filas definidas en relación a R (en la ordenación de las filas de la partición de R). Esta cláusula especifica el grupo de agregación. Si no se especifica esta cláusula, el valor por omisión es el mismo que RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, lo que proporciona un resultado de agregación acumulativo.

ROWS
Indica que el grupo de agregación se define mediante el contaje de filas.

RANGE
Indica que el grupo de agregación se define mediante un valor de desplazamiento con respecto a una clave de clasificación.

inicio-grupo
Especifica el punto de inicio del grupo de agregación. El final del grupo de agrupación es la fila actual. La cláusula inicio-grupo es equivalente a una cláusula entre-grupo en la forma ″BETWEEN inicio-grupo AND CURRENT ROW″.

entre-grupo
Especifica el inicio y final del grupo de agregación basándose en ROWS o RANGE.

final-grupo
Especifica el punto final del grupo de agregación. El inicio del grupo de agregación es la fila actual. La especificación de la cláusula final-grupo es equivalente a la de una cláusula entre-grupo del formato ″BETWEEN CURRENT ROW AND final-grupo″.

UNBOUNDED PRECEDING
Incluye la partición completa que precede a la fila actual. Esto se puede especificar con ROWS o RANGE. También se puede especificar con varias expresiones-clave-clasificación en la cláusula-orden-ventana.

UNBOUNDED FOLLOWING
Incluye la partición completa que sigue a la fila actual. Esto se puede especificar con ROWS o RANGE. También se puede especificar con varias expresiones-clave-clasificación en la cláusula-orden-ventana.

CURRENT ROW
Especifica el inicio o el final del grupo de agregación basándose en la fila actual. Si se especifica ROWS, la fila actual es el límite del grupo de agregación. Si se especifica RANGE, el límite del grupo de agregación incluye el conjunto de filas con los mismos valores para las *expresiones-clave-clasificación* que la fila actual. Esta cláusula no se puede especificar en *límite-grupo2* si *límite-grupo1* especifica el *valor* FOLLOWING.

valor **PRECEDING**
Especifica el rango o número de filas que preceden a la fila actual. Si se especifica ROWS, *valor* es un entero positivo que indica un número de filas. Si se especifica RANGE, el tipo de datos de *valor* debe ser comparable con el tipo de la expresión-clave-clasificación de la cláusula-orden-ventana. Sólo puede haber una sola expresión-clave-clasificación y el tipo de datos de esa expresión debe permitir la operación de resta. Esta cláusula no se puede especificar en *límite-grupo2* si *límite-grupo1* es CURRENT ROW o *valor* FOLLOWING.

valor **FOLLOWING**
Especifica el rango o número de filas que siguen a la fila actual. Si se especifica ROWS, *valor* es un entero positivo que indica un número de filas. Si se especifica RANGE, el tipo de datos de *valor* debe ser comparable con el tipo de la expresión-clave-clasificación de la cláusula-orden-ventana. Sólo puede haber una sola expresión-clave-clasificación y el tipo de datos de esa expresión debe permitir la operación de suma.

Ejemplos:

- Este ejemplo muestra la ordenación de los empleados, dispuestos por apellidos, de acuerdo con un salario total (salario más prima) que sea mayor que $30.000.

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
    RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
  FROM EMPLOYEE WHERE SALARY+BONUS > 30000
  ORDER BY LASTNAME
```

  Observe que si el resultado debe estar ordenado de acuerdo con la escala de salarios, debe sustituir ORDER BY LASTNAME por:

```
ORDER BY RANK_SALARY
```

  o

```
ORDER BY RANK() OVER (ORDER BY SALARY+BONUS DESC)
```

- Este ejemplo ordena los departamentos de acuerdo con su salario total medio.

```
SELECT WORKDEPT, AVG(SALARY+BONUS) AS AVG_TOTAL_SALARY,
    RANK() OVER (ORDER BY AVG(SALARY+BONUS) DESC) AS RANK_AVG_SAL
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  ORDER BY RANK_AVG_SAL
```

- Este ejemplo ordena los empleados de un departamento de acuerdo con su nivel de formación. Si varios empleados de un departamento tienen el mismo nivel, ello no debe suponer un aumento del nivel siguiente.

```
SELECT WORKDEPT, EMPNO, LASTNAME, FIRSTNME, EDLEVEL,
    DENSE_RANK() OVER
      (PARTITION BY WORKDEPT ORDER BY EDLEVEL DESC) AS RANK_EDLEVEL
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

- Este ejemplo proporciona números a las filas del resultado de una consulta.

```
SELECT ROW_NUMBER() OVER (ORDER BY WORKDEPT, LASTNAME) AS NUMBER,
    LASTNAME, SALARY
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

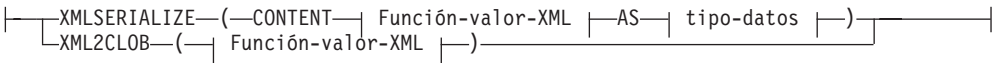
- Este ejemplo lista los cinco empleados con mayores ingresos.

```
SELECT EMPNO, LASTNAME, FIRSTNME, TOTAL_SALARY, RANK_SALARY
  FROM (SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
    RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
    FROM EMPLOYEE) AS RANKED_EMPLOYEE
  WHERE RANK_SALARY < 6
  ORDER BY RANK_SALARY
```

Observe que primero se ha utilizado una expresión de tabla anidada para calcular el resultado, incluidos los niveles de ordenación, para poder utilizar el nivel en la cláusula WHERE. También se podría haber utilizado una expresión de tabla común.

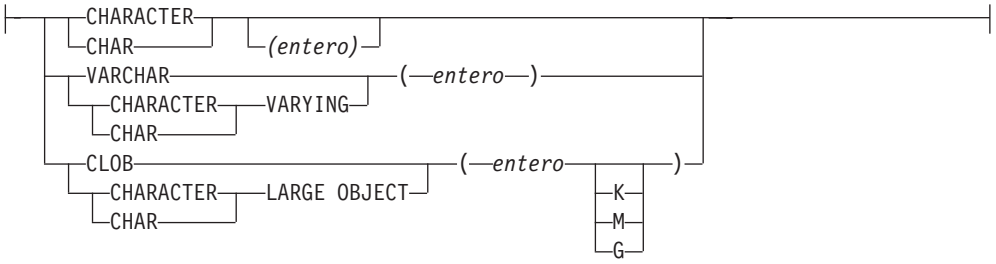
Funciones XML

Función-XML:

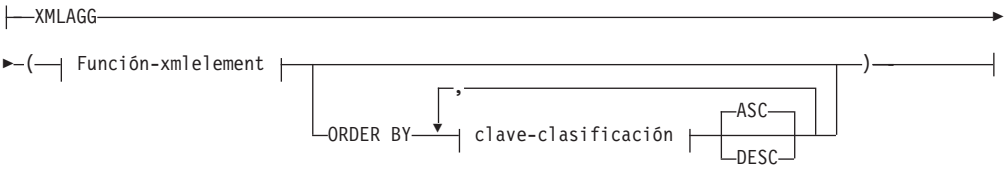


Función-valor-XML:

tipo-datos:

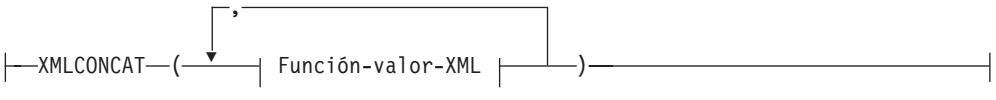


Función-xmlagg:

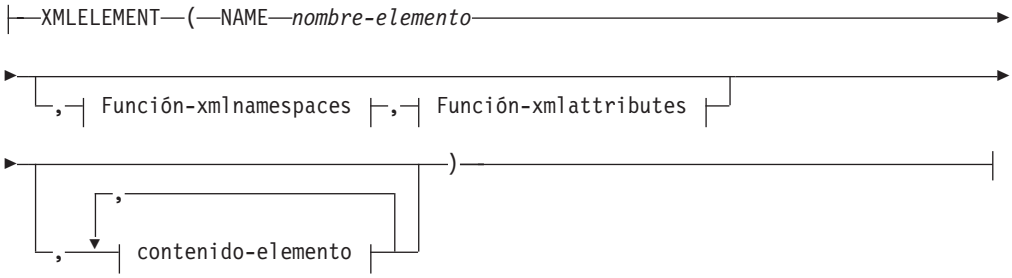


clave-clasificación:

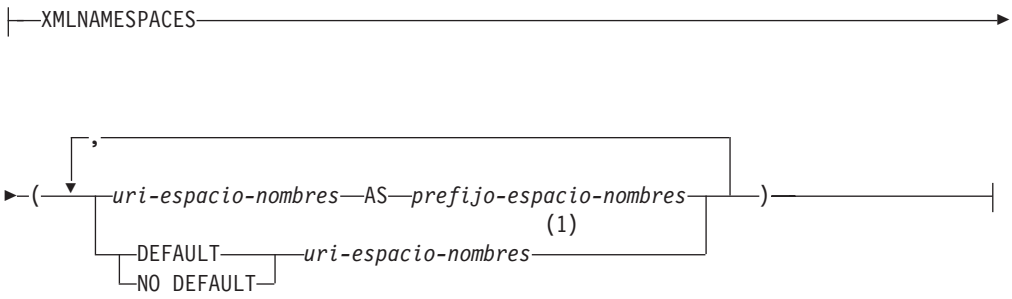
Función-xmlconcat:



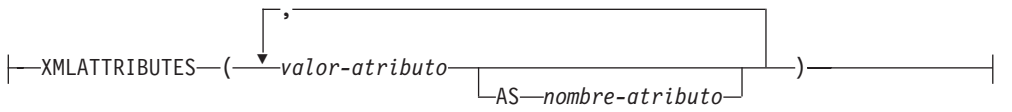
Función-xmlelement:



Función-xmlnamespaces:

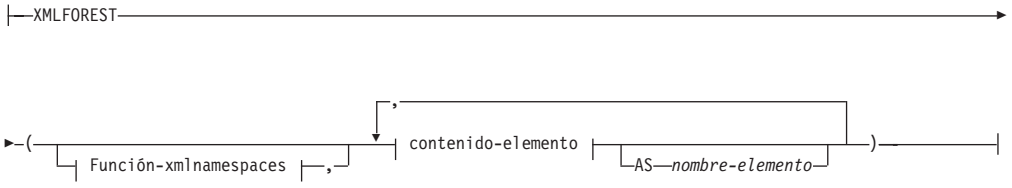


Función-xmlattributes:



contenido-elemento:

Función-xmlforest:



Notas:

1 Sólo puede especificarse DEFAULT o NO DEFAULT una vez en los argumentos de XMLNAMESPACES.

XMLSERIALIZE

Devuelve el argumento como un valor del tipo de datos especificado como parte de los argumentos de función. El esquema es SYSIBM. El argumento

debe ser una expresión del tipo de datos XML. El resultado tiene el tipo de datos especificado por el usuario. Si el resultado de la *función-valor-XML* es nulo, el resultado es el valor nulo.

CONTENT
Especifica que el valor de *función-valor-XML* puede estar formado por más de un elemento de nivel superior.

AS *tipo-datos*
Los tipos de datos soportados son CHAR, VARCHAR y CLOB. Si no se especifica ninguna longitud para CHAR, el valor por omisión de la longitud es 1. Es necesario especificar el valor de la longitud para VARCHAR. Si no se especifica ninguna longitud para CLOB, el valor por omisión de la longitud es 1M.

XML2CLOB
Devuelve el argumento como un valor CLOB. El esquema es SYSIBM. El argumento debe ser una expresión del tipo de datos XML. El resultado tiene el tipo de datos CLOB.

Se proporciona la función XML2CLOB para mantener la compatibilidad con los releases anteriores. En este caso, debe utilizarse la función XMLSERIALIZE en su lugar.

XMLAGG
Devuelve la concatenación de un conjunto de valores de XML. El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado. El tipo de datos del resultado es XML y su longitud está establecida en 1 073 741 823. Si la función XMLAGG se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la concatenación de los valores del conjunto.

ORDER BY
Especifica el orden de las filas del mismo conjunto de agrupación que se procesan en la agregación. Si se omite la cláusula ORDER BY o si ésta no puede distinguir el orden de los datos de la columna, las filas del mismo conjunto de agrupación se ordenan de forma arbitraria.

clave-clasificación
La clave de clasificación puede ser un nombre de columna o una expresión-clave-clasificación. Observe que si la clave de clasificación es una constante, no hace referencia a la posición de la columna de salida(como en la cláusula ORDER BY normal) sino que es simplemente una constante, que no implica ninguna clave de clasificación.

Las restricciones respecto a la utilización de la función XMLAGG son las siguientes:

- Las funciones de columna no pueden utilizarse como entrada directa (SQLSTATE 42607).
- XMLAGG no puede utilizarse como función de columna de una función agregada OLAP (SQLSTATE 42601).

XMLCONCAT
Devuelve la concatenación de un número variable de argumentos de XML. El esquema es SYSIBM. Los argumentos deben ser expresiones del tipo de datos XML. El resultado tiene el mismo tipo de datos XML interno que los argumentos. Un valor nulo devuelto desde un argumento de entrada de la función XMLCONCAT se pasa por alto. Si todos los argumentos de

XMLCONCAT son nulos, el resultado es el valor nulo. De lo contrario, el resultado es la concatenación de los valores de XML.

XMLELEMENT
Construye un elemento XML a partir de los argumentos. El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado. Esta función toma un nombre de elemento, un conjunto opcional de declaraciones de espacios de nombres, un conjunto opcional de atributos y cero o más argumentos que formarán el contenido del elemento. Si el contenido del elemento es nulo, el resultado será un elemento vacío. El tipo de datos del resultado es XML.

NAME
Esta palabra clave precede el nombre de un elemento XML.

nombre-elemento
El nombre del elemento XML. Debería ser un nombre XML QName. Para obtener más información sobre los nombres válidos, vea las especificaciones sobre espacios de nombres W3C XML .

Función-xmlnamespaces
Declaraciones de espacios de nombres XML que son el resultado de la función XMLNAMESPACES.

Función-xmlattributes
Atributos XML que son el resultado de la función XMLATTRIBUTES.

contenido-elemento
El contenido de los elementos generados se especifica mediante una expresión o una lista de expresiones. El tipo de datos del resultado de la expresión debe ser SMALLINT, INTEGER, BIGINT, DECIMAL, NUMERIC, REAL, DOUBLE, CHAR, VARCHAR, LONG VARCHAR, CLOB, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, DATE, TIME, TIMESTAMP, XML cualquier tipo diferenciado cuyo tipo fuente sea uno de los tipos de datos anteriores. No se permiten datos de serie de caracteres que estén definidos como FOR BIT DATA. Las expresiones pueden ser cualquier expresión de SQL, pero no pueden incluir una selección completa escalar ni una subconsulta.

XMLATTRIBUTES
Construye los atributos XML a partir de los argumentos. El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado. El resultado tiene el mismo tipo de datos XML interno que los argumentos.

valor-atributo
El valor de atributo es una expresión. El tipo de datos del resultado de la expresión debe ser SMALLINT, INTEGER, BIGINT, DECIMAL, NUMERIC, REAL, DOUBLE, CHAR, VARCHAR, LONG VARCHAR, CLOB, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, DATE, TIME, TIMESTAMP cualquier tipo diferenciado cuyo tipo fuente sea uno de los tipos de datos anteriores. No se permiten datos de serie de caracteres que estén definidos como FOR BIT DATA. La expresión puede ser cualquier expresión de SQL pero no puede incluir una selección completa escalar ni una subconsulta. Si la expresión no es una referencia de columna simple, debe especificarse un nombre de atributo. No se permiten los nombres de atributos duplicados (SQLSTATE 42713).

nombre-atributo
El nombre de atributo es un identificador SQL.Debe tener el formato de un

nombre calificado XML o QName (SQLSTATE 42634). Para obtener más información sobre los nombres válidos, vea las especificaciones sobre espacios de nombres W3C XML . El nombre del atributo no puede ser `xmlns` ni llevar el prefijo `xmlns:`. Para declarar un espacio de nombres, debe utilizarse la función XMLNAMESPACES.

XMLFOREST
Construye una secuencia (bosque) de elementos XML a partir de los argumentos. El esquema es SYSIBM. El resultado tiene el mismo tipo de datos XML interno que los argumentos. Un valor nulo devuelto desde un argumento de entrada de la función XMLFOREST se pasa por alto (no se genera ningún elemento). Si todos los argumentos de XMLFOREST son nulos, el resultado es el valor nulo. De lo contrario, el resultado es la secuencia de elementos XML devueltos.

Función-xmlnamespaces
Declaraciones de espacios de nombres XML que son el resultado de la función XMLNAMESPACES.

contenido-elemento
El contenido del elemento es una expresión. El tipo de datos del resultado de la expresión debe ser SMALLINT, INTEGER, BIGINT, DECIMAL, NUMERIC, REAL, DOUBLE, CHAR, VARCHAR, LONG VARCHAR, CLOB, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, DATE, TIME, TIMESTAMP cualquier tipo diferenciado cuyo tipo fuente sea uno de los tipos de datos anteriores. No se permiten datos de serie de caracteres que estén definidos como FOR BIT DATA. La expresión puede ser cualquier expresión de SQL pero no puede incluir una selección completa escalar ni una subconsulta. Si la expresión no es una referencia de columna simple, debe especificarse un nombre de elemento.

nombre-elemento
El nombre de elemento es un identificador SQL.El nombre del elemento debe tener el formato de un nombre calificado XML o QName (SQLSTATE 42634). Para obtener más información sobre los nombres válidos, vea las especificaciones sobre espacios de nombres W3C XML .

XMLNAMESPACES
Construye las declaraciones de espacios de nombres XML a partir de los argumentos. El esquema es SYSIBM. El tipo de datos del resultado es XML. Las declaraciones de espacios de nombres están en el ámbito de los elementos generados por las funciones XMLELEMENT y XMLFOREST. Es posible utilizar el prefijo predefinido para el espacio de nombres, `xml`, sin necesidad de definirlo de forma explícita. El prefijo `xml` no puede redefinirse.

uri-espacio-nombres
El identificador de recursos universal (URI) de los espacios de nombre es una constante de serie de caracteres. Esta constante de serie de caracteres no puede estar vacía.

prefijo-espacio-nombres
El prefijo de espacio de nombres es un identificador SQL. Debe tener el formato de un nombre XML NCName (SQLSTATE 42635). Para obtener más información sobre los nombres válidos, vea las especificaciones sobre espacios de nombres W3C XML. El prefijo no puede ser `xml` ni `xmlns`. El prefijo debe ser exclusivo dentro de la lista de declaraciones de espacios de nombres.

DEFAULT *uri-espacio-nombres*
Especifica el espacio de nombres por omisión que debe utilizarse para este

elemento XML y para todos los elementos XML incluidos en este elemento XML, a excepción de aquellas *funciones-xmlnamespaces* que especifiquen NO DEFAULT para sus espacios de nombres.

NO DEFAULT
Especifica que no debe utilizarse ningún espacio de nombres por omisión para este elemento XML ni para ninguno de los elementos XML incluidos en este elemento XML, a excepción de aquellas *funciones-xmlnamespaces* que especifiquen DEFAULT para sus espacios de nombres.

Ejemplos:

- Construya un valor CLOB a partir de la expresión devuelta por la función XMLELEMENT. La consulta

```
SELECT e.empno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "Emp", e.firstnme || ' ' ||
    e.lastname) AS CLOB)
  AS "Result" FROM employee e
  WHERE e.edlevel = 12
```

  genera el resultado siguiente:

```
EMPNO      Result
 000290     <Emp>JOHN PARKER</Emp>
 000310     <Emp>MAUDE SETRIGHT</Emp>
```

- Genere un elemento departamental (para cada departamento) con una lista de empleados, clasificados por el apellido del empleado:

```
SELECT XMLSERIALIZE(CONTENT XMLELEMENT(NAME "Department",
                            XMLATTRIBUTES(e.workdept AS "name"),
                            XMLAGG(XMLELEMENT(NAME "emp", e.lastname)
                               ORDER BY e.lastname
                                 )
                           ) AS CLOB ) AS "dept_list"
  FROM employee e
  WHERE e.workdept IN ('C01','E21')
  GROUP BY workdept
```

  Esta consulta genera la salida siguiente. Observe que en realidad no se genera ningún espacio ni carácter de salto de línea en el resultado; el resultado siguiente se ha formateado para proporcionar mayor claridad.

```
dept_list
<Department name = "C01">
    <emp>KWAN</emp>
    <emp>NICHOLLS</emp>
    <emp>QUINTANA</emp>
</Department>
<Department name = "E21">
    <emp>GOUNOT</emp>
    <emp>LEE</emp>
    <emp>MEHTA</emp>
    <emp>SPENSER</emp>
</Department>
```

- Para cada departamento que informe al departamento A00, cree un elemento XML vacío denominado Mgr con un atributo de ID igual al MGRNO. La consulta

```
SELECT d.deptno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "Mgr",
  XMLATTRIBUTES(d.mgrno AS "ID")) AS CLOB)
  AS "Result" FROM department d
  WHERE d.admrdept = 'A00'
```

  genera el resultado siguiente:

```
DEPTNO     Result
A00        <Mgr ID="000010"/>
B01        <Mgr ID="000020"/>
C01        <Mgr ID="000030"/>
D01        <Mgr/>
E01        <Mgr ID="000050">
```

- Genere un elemento de XML denominado Emp para cada empleado, con elementos anidados para el nombre completo del empleado y la fecha en que fue contratado. La consulta

```
SELECT e.empno, XMLSERIALIZE
 (CONTENT XMLELEMENT(NAME "Emp",
  XMLELEMENT(NAME "name", e.firstnme || ' ' || e.lastname),
  XMLELEMENT(NAME "hiredate", e.hiredate)) AS CLOB)
  AS "Result" FROM employee e
  WHERE e.edlevel = 12
```

  genera el resultado siguiente (formateado aquí para su conveniencia; el fragmento del XML de salida no tiene caracteres de espacios en blanco ajenos):

```
EMPNO      Result
000290     <Emp>
           <name>JOHN PARKER</name>
           <hiredate>1980-05-30</hiredate>
           </Emp>
000310     <Emp>
           <name>MAUDE SETRIGHT</name>
           <hiredate>1964-09-12</hiredate>
           </Emp>
```

- Utilizando la función XMLATTRIBUTES, junto con las funciones XMLSERIALIZE y XMLELEMENT, construya los atributos XML. La consulta

```
SELECT XMLSERIALIZE(CONTENT XMLELEMENT(NAME "Emp_Exempt",
  XMLATTRIBUTES(e.firstnme, e.lastname AS "name:last", e.midinit)) AS CLOB)
    AS "Result"
  FROM employee e
  WHERE e.lastname='GEYER'
```

  genera el resultado siguiente:

```
<Emp_Exempt
   FIRSTNME="JOHN"
   name_last="GEYER"
   MIDINIT="B">
</Emp_Exempt>
```

- Genere un elemento Emp para cada empleado cualificado que contenga una secuencia de subelementos generados a partir de argumentos de XMLFOREST. La consulta

```
SELECT e.empno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "Emp",
    XMLFOREST(e.firstnme || ' ' || e.lastname AS "Name" , e.hiredate)) AS CLOB)
  AS "Result" FROM employee e
  WHERE e.edlevel = 12
```

  genera el resultado siguiente (formateado aquí para su conveniencia; el XML de salida no tiene caracteres de espacios en blanco ajenos):

```
EMPNO      Result
 000290     <Emp>
              <Name>JOHN PARKER</Name>
              <HIDREDATE>1980-05-30</HIREDATE>
            </Emp>
 000310     <Emp>
              <Name>MAUDE SETRIGHT</Name>
              <HIREDATE>1964-09-12</HIREDATE>
            </Emp>
```

- Genere el apellido y el nombre de cada empleado cualificado. La consulta

```
SELECT e.empno, XMLSERIALIZE(CONTENT XMLCONCAT(XMLELEMENT(NAME "firstname",
      e.firstnme),
    XMLELEMENT(NAME "lastname", e.lastname)) AS CLOB)
  AS "Result" FROM employee e
  WHERE e.edlevel = 12
```

  genera el resultado siguiente:

```
EMPNO      Result
 000290     <firstname>JOHN</firstname><lastname>PARKER</lastname>
 000310     <firstname>MAUDE</firstname><lastname>SETRIGHT</lastname>
```

- Genere un elemento XML denominado `adm:employee` y un atributo XML `adm:department`, ambos asociados con un espacio de nombres cuyo prefijo es `adm`.

```
SELECT empno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "adm:employee",
  XMLNAMESPACES('http://www.adm.com' AS "adm"),
  XMLATTRIBUTES(workdept AS "adm:department"), lastname) AS CLOB) AS "Result"
  FROM employee
  WHERE job = 'ANALYST'
```

  Esta consulta genera un valor XML con una representación textual, similar al mostrado en la salida siguiente (formateado aquí para su conveniencia; el fragmento del XML de salida no tiene caracteres de espacios en blanco ajenos y la salida generalmente aparece como una línea):

```
  EMPNO      Result
000130 <adm:employee xmlns:adm="http://www.adm.com"
                     adm:department="C01">QUINTANA
       </adm:employee>
000140 <adm:employee xmlns:adm="http://www.adm.com"
                     adm:department="C01">NICHOLLS
       </adm:employee>
```

- Genere una secuencia de elementos generados a partir de los argumentos de XMLFOREST. Declare también un espacio de nombres por omisión para asociarlo al primer elemento y un espacio de nombres cuyo prefijo sea `d` asociado al segundo elemento.

```
SELECT empno, XMLSERIALIZE(CONTENT XMLFOREST
  (XMLNAMESPACES(DEFAULT 'http://hr.org',
    'http://fed.gov' AS "d"),
    lastname, job AS "d:job") AS CLOB) AS "Result"
  FROM employee
  WHERE edlevel = 12
```

  Esta consulta genera un valor XML con una representación textual, similar al mostrado en la salida siguiente (formateado aquí para su conveniencia; el fragmento del XML de salida no tiene caracteres de espacios en blanco ajenos y la salida generalmente aparece como una línea):

```
  EMPNO      Result
000290 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">PARKER
       </LASTNAME>
       <d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR
       </d:job>
000310 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">SETRIGHT
       </LASTNAME>
       <d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR
       </d:job>
```

- Genere un elemento XML denominado ″employee″ asociado con un espacio de nombres por omisión y un subelemento denominado ″department″ que no utilice el espacio de nombres por omisión.

```
SELECT emp.empno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "employee",
  XMLNAMESPACES(DEFAULT 'http://hr.org'), emp.lastname,
  XMLELEMENT(NAME "department",
    XMLNAMESPACES(NO DEFAULT),
      emp.workdept)) AS CLOB) AS "Result"
  FROM employee emp
  WHERE emp.edlevel = 12
```

Esta consulta genera un valor XML serializado de forma similar a como aparece en la salida siguiente (formateado aquí para su conveniencia; el fragmento del XML de salida no tiene caracteres de espacios en blanco ajenos y la salida generalmente aparece como una línea):

```
  EMPNO      Result
000290 <employee xmlns="http://hr.org">PARKER
              <department xmlns="">E11
              </department>
       </employee>
000310 <employee xmlns="http://hr.org">SETRIGHT
              <department xmlns="">E11
              </department>
       </employee>
```

- Genere un elemento XML denominado ″employee″ asociado con un espacio de nombres por omisión y un subelemento denominado ″job″ que no utilice el espacio de nombres por omisión y cuyo subelemento ″department″ utilice un espacio de nombres por omisión.

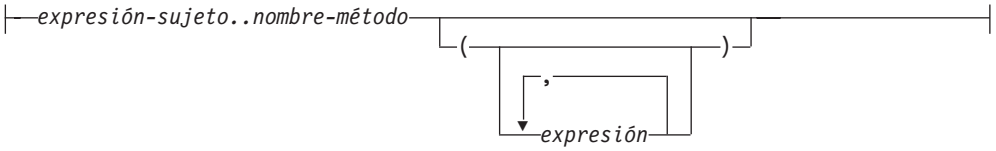
```
SELECT emp.empno, XMLSERIALIZE(CONTENT XMLELEMENT(NAME "employee",
  XMLNAMESPACES(DEFAULT 'http://hr.org'), emp.lastname,
  XMLELEMENT(NAME "job",
    XMLNAMESPACES(NO DEFAULT), emp.job,
    XMLELEMENT(NAME "department",
      XMLNAMESPACES(DEFAULT 'http://adm.org'), emp.workdept))) AS CLOB) AS "Result"
  FROM employee emp
  WHERE emp.edlevel = 12
```

Esta consulta genera un valor XML, cuyo formato serializado se parece al resultado siguiente (formateado aquí para su conveniencia; el fragmento del XML de salida no tiene caracteres de espacios en blanco ajenos y la salida generalmente aparece como una línea):

```
  EMPNO      Result
000290 <employee xmlns="http://hr.org">PARKER
           <job xmlns="">OPERATOR
              <department xmlns="http://adm.org">E11
              </department>
           </job>
       </employee>
000310 <employee xmlns="http://hr.org">SETRIGHT
           <job xmlns="">OPERATOR
              <department xmlns="http://adm.org">E11
              </department>
           </job>
       </employee>
```

Invocación de métodos

invocación-método:



El método observador y el método mutador, ambos generados por el sistema, así como los métodos definidos por el usuario se invocan utilizando el operador formado por dos puntos.

expresión-sujeto
Es una expresión con un tipo resultante estático que es un tipo estructurado definido por el usuario.

nombre-método
Es el nombre no calificado de un método. El tipo estático de *expresión-sujeto* o uno de sus supertipos debe incluir un método que tenga el nombre especificado.

(*expresión***,...)**
Los argumentos de *nombre-método* se especifican entre paréntesis. Se pueden utilizar paréntesis vacíos para indicar que no existen argumentos. El *nombre-método* y los tipos de datos de las expresiones argumento especificadas se utilizan para obtener el método específico, basándose en el tipo estático de *expresión-sujeto*.

El operador .. utilizado para invocar el método es un operador infijo que define una prioridad de operaciones de izquierda a derecha. Por ejemplo, las dos expresiones siguientes son equivalentes:

```
a..b..c + x..y..z
```

y

```
((a..b)..c) + ((x..y)..z)
```

Si un método no tiene ningún otro parámetro que no sea su sujeto, éste se puede invocar con o sin paréntesis. Por ejemplo, las dos expresiones siguientes son equivalentes:

```
point1..xpoint1..x()
```

Los sujetos nulos de una invocación de método se gestionan de este modo:

- Si un método mutador generado por el sistema se invoca con un sujeto nulo, se produce un error (SQLSTATE 2202D)
- Si cualquier método distinto de un método mutador generado por el sistema se invoca con un sujeto nulo, el método no se ejecuta y su resultado es nulo. Esta regla incluye los métodos definidos por el usuario con SELF AS RESULT.

Cuando se crea un objeto de base de datos (por ejemplo, un paquete, vista o activador), se determina el método de ajuste óptimo que existe para cada invocación de método.

Nota: Los métodos de los tipos definidos con WITH FUNCTION ACCESS también se pueden invocar utilizando la notación normal de funciones. La resolución de la función considera como aceptables todas las funciones, así como los métodos con acceso a función. Sin embargo, las funciones no se pueden invocar utilizando la invocación de método. La resolución del método considera aceptables todos los métodos, pero no las funciones. Si la resolución no proporciona una función o método apropiado, se produce un error (SQLSTATE 42884).

Ejemplo:

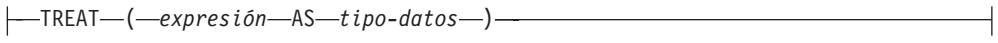
- Este ejemplo utiliza el operador .. para invocar un método llamado AREA. Se supone que existe una tabla llamada RINGS, con una columna CIRCLE_COL del

tipo estructurado CIRCLE. Se supone también que el método AREA se ha definido previamente para el tipo CIRCLE como `AREA() RETURNS DOUBLE`.

```
SELECT CIRCLE_COL..AREA() FROM RINGS
```

Tratamiento de los subtipos

tratamiento-subtipo:



El *tratamiento-subtipo* se utiliza para convertir una expresión de tipo estructurado en uno de sus subtipos. El tipo estático de *expresión* debe ser un tipo estructurado definido por el usuario, y ese tipo debe ser el mismo que *tipo-datos* o que un subtipo de él. Si el nombre de tipo especificado en *tipo-datos* no está calificado, se utiliza la vía de acceso de SQL para resolver la referencia al tipo. El tipo estático del resultado de tratamiento-subtipo es *tipo-datos*, y el valor del tratamiento-subtipo es el valor de la expresión. Durante la ejecución, si el tipo dinámico de la expresión no es *tipo-datos* o un subtipo de *tipo-datos*, se produce un error (SQLSTATE 0D000).

Ejemplo:

- En este ejemplo, todas las instancias de objetos de la columna CIRCLE_COL están definidas con el tipo dinámico COLOREDCIRCLE para una aplicación. Se utiliza la consulta siguiente para invocar el método RGB para tales objetos. Se supone que existe una tabla llamada RINGS, con una columna CIRCLE_COL del tipo estructurado CIRCLE. Se supone también que COLOREDCIRCLE es un subtipo de CIRCLE y que el método RGB se ha definido previamente para COLOREDCIRCLE como `RGB() RETURNS DOUBLE`.

  ```
  SELECT TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
     FROM RINGS
  ```

  Durante la ejecución, si hay instancias del tipo dinámico CIRCLE, se produce un error (SQLSTATE 0D000). Este error se puede evitar utilizando el predicado TYPE en una expresión CASE, del modo siguiente:

  ```
  SELECT (CASE
     WHEN CIRCLE_COL IS OF (COLOREDCIRCLE)
      THEN TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
        ELSE NULL
          END)      FROM RINGS
  ```

Referencia de secuencia

referencia-secuencia:

expresión-nextval:

expresión-prevval:

NEXT VALUE FOR *nombre-secuencia*
Una expresión NEXT VALUE genera y devuelve el siguiente valor de la secuencia especificada por *nombre-secuencia*.

PREVIOUS VALUE FOR *nombre-secuencia*
Una expresión PREVIOUS VALUE devuelve el valor generado más recientemente de la secuencia especificada para una sentencia anterior del proceso de aplicación actual. Se puede hacer referencia a este valor repetidamente utilizando expresiones PREVIOUS VALUE que especifican el nombre de la secuencia. Pueden existir múltiples instancias de las expresiones PREVIOUS VALUE especificando el mismo nombre de secuencia en una sola sentencia; todas ellas devuelven el mismo valor. En un entorno de bases de datos particionadas, es posible que una expresión PREVIOUS VALUE no devuelva el valor generado más recientemente.

Una expresión PREVIOUS VALUE sólo se puede utilizar si ya se ha hecho referencia a una expresión NEXT VALUE que especifica el mismo nombre de secuencia en el proceso de aplicación actual, ya sea en la transacción actual ya sea en una transacción anterior (SQLSTATE 51035).

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NEXTVAL y PREVVAL pueden especificarse en lugar de NEXT VALUE y PREVIOUS VALUE.
- Se genera un valor nuevo para una secuencia cuando la expresión NEXT VALUE especifica el nombre de dicha secuencia. Sin embargo, si existen múltiples instancias de una expresión NEXT VALUE que especifican el mismo nombre de secuencia en una consulta, el contador para la secuencia se incrementa sólo una vez para cada fila del resultado y todas las instancias de NEXT VALUE devuelven el mismo valor para una fila del resultado.
- Se puede utilizar el mismo número de secuencia como valor de clave de unicidad en dos tablas independientes haciendo referencia al número de secuencia con una expresión NEXT VALUE para la primera fila (esto genera el valor de secuencia) y una expresión PREVIOUS VALUE para las demás filas (la instancia de PREVIOUS VALUE hace referencia al valor de secuencia generado más recientemente en la sesión actual), tal como se muestra a continuación:

```
INSERT INTO order(orderno, cutno)
  VALUES (NEXT VALUE FOR order_seq, 123456);

INSERT INTO line_item  (orderno, partno, quantity)
  VALUES (PREVIOUS VALUE FOR order_seq, 987654, 1);
```

- Las expresiones NEXT VALUE y PREVIOUS VALUE pueden especificarse en los lugares siguientes:
  - sentencia-select o sentencia SELECT INTO (en la cláusula-select, a condición de que la sentencia no contenga una palabra clave DISTINCT, una cláusula GROUP BY, una cláusula ORDER BY, una palabra clave UNION, una palabra clave INTERSECT o una palabra clave EXCEPT)
  - sentencia INSERT (en una cláusula VALUES)
  - sentencia INSERT (en la cláusula-select de la selección completa (fullselect))
  - sentencia UPDATE (en la cláusula SET (una sentencia UPDATE buscada o colocada), excepto que no se puede especificar NEXT VALUE en la cláusula-select de la selección completa de una expresión de la cláusula SET)

  - setencia SET variable (excepto en la cláusula-select de la selección completa de una expresión; una expresión NEXT VALUE puede especificarse en un activador, pero una expresión PREVIOUS VALUE no puede especificarse)
  - sentencia VALUES INTO (en la cláusula-select de la selección completa (fullselect) de una expresión)
  - sentencia CREATE PROCEDURE (en el cuerpo-rutina de un procedimiento SQL)
  - sentencia CREATE TRIGGER en la acción-activada (se puede especificar una expresión NEXT VALUE, pero no se puede especificar una expresión PREVIOUS VALUE)
- Las expresiones NEXT VALUE y PREVIOUS VALUE no se pueden especificar (SQLSTATE 428F9) en los lugares siguientes:
  - las condiciones de unión de una unión externa completa
  - el valor DEFAULT de una columna en una sentencia CREATE o ALTER TABLE
  - la definición de columna generada en una sentencia CREATE o ALTER TABLE
  - la definición de tabla de resumen de una sentencia CREATE TABLE o ALTER TABLE
  - la condición de una restricción CHECK
  - sentencia CREATE TRIGGER (se puede especificar una expresión NEXT VALUE, pero no se puede especificar una expresión PREVIOUS VALUE)
  - Sentencia CREATE VIEW
  - Sentencia CREATE METHOD
  - Sentencia CREATE FUNCTION
- Además, no se puede especificar una expresión NEXT VALUE (SQLSTATE 428F9) en los lugares siguientes:
  - la expresión CASE
  - la lista de parámetros de una función agregada
  - la subconsulta en un contexto distinto de los explícitamente permitidos mencionados anteriormente
  - la sentencia SELECT para la que la SELECT externa contiene un operador DISTINCT
  - la condición de unión de una unión
  - la sentencia SELECT para la que la SELECT externa contiene una cláusula GROUP BY
  - la sentencia SELECT para la que la SELECT externa está combinada con otra sentencia SELECT utilizando el operador establecido UNION, INTERSECT o EXCEPT
  - la expresión de tabla anidada
  - la lista de parámetros de una función de tabla
  - la cláusula WHERE de la sentencia SELECT más externa o una sentencia DELETE o UPDATE
  - la cláusula ORDER BY de la sentencia SELECT más externa
  - la cláusula-select de la selección completa (fullselect) de una expresión, en la cláusula SET de una sentencia UPDATE
  - la sentencia IF, WHILE, DO ... UNTIL o CASE de una rutina SQL

- Cuando se genera un valor para una secuencia, se consume dicho valor y, la siguiente vez que se solicita un valor, se genera un valor nuevo. Esto es válido incluso cuando la sentencia que contiene la expresión NEXT VALUE falla o se retrotrae.

  Si una sentencia INSERT incluye una expresión NEXT VALUE en la lista VALUES para la columna y si se produce un error en algún punto durante la ejecución de INSERT (puede ser un problema al generar el siguiente valor de secuencia o un problema con el valor de otra columna), se produce una anomalía de inserción (SQLSTATE 23505) y se considera que el valor generado para la secuencia se ha consumido. En algunos casos, al volver a emitir la misma sentencia INSERT se puede obtener un resultado satisfactorio.

  Por ejemplo, considere un error que es el resultado de la existencia de un índice de unicidad para la columna para la que se ha utilizado NEXT VALUE y el valor de secuencia generado ya existe en el índice. Es posible que el siguiente valor generado para la secuencia sea un valor que no existe en el índice y, por consiguiente, el INSERT subsiguiente dará un resultado satisfactorio.
- Si al generar un valor para un secuencia, se excede el valor máximo para la secuencia (o el valor mínimo para una secuencia descendente) y no se permiten ciclos, se producirá un error (SQLSTATE 23522). En este caso, el usuario puede modificar (ALTER) la secuencia para ampliar el rango de valores aceptables, habilitar ciclos para la secuencia o eliminar (DROP) la secuencia y crear (CREATE) una nueva con un tipo de datos diferente que tenga un mayor rango de valores.

  Por ejemplo, una secuencia puede haberse definido con un tipo de datos de SMALLINT y, finalmente, la secuencia se queda sin valores asignables. Elimine (DROP) y vuelva a crear la secuencia con la nueva definición para volver a definir la secuencia como INTEGER.
- Una referencia a una expresión NEXT VALUE en la sentencia de selección (select) de un cursor hace referencia a un valor que se genera para una fila de la tabla resultante. Se genera un valor de secuencia para una expresión NEXT VALUE para cada fila que se busca desde la base de datos. Si se realiza el bloqueo en el cliente, puede que los valores se hayan generado en el servidor antes del proceso de la sentencia FETCH. Esto puede producirse cuando existe bloqueo de las filas de la tabla resultante. Si la aplicación cliente no capta (FETCH) explícitamente todas las filas que la base de datos ha materializado, la aplicación no verá los resultados de todos los valores de secuencia generados (para las filas materializadas que no se ha devuelto).
- Una referencia a una expresión PREVIOUS VALUE de la sentencia de selección (select) de un cursor hace referencia a un valor que se ha generado para la secuencia especificada antes de la apertura del cursor. Sin embargo, el cierre del cursor puede afectar a los valores devueltos por PREVIOUS VALUE para la secuencia especificada en las sentencias futuras o incluso para la misma sentencia en el caso de que se vuelva a abrir el cursor. Esto sucederá cuando la sentencia de selección del cursor incluya una referencia a NEXT VALUE para el mismo nombre de secuencia.

Ejemplos:

Supongamos que existe una tabla llamada ″order″ y que se crea una secuencia llamada ″order_seq″ del modo siguiente:

```
CREATE SEQUENCE order_seq
   START WITH 1
   INCREMENT BY 1
NO MAXVALUE
NO CYCLE
  CACHE 24
```

A continuación se muestran algunos ejemplos de cómo generar un número de secuencia ″order_seq″ con una expresión NEXT VALUE:

```
INSERT INTO order(orderno, custno)
  VALUES (NEXT VALUE FOR order_seq, 123456);
```

o

```
UPDATE order
  SET orderno = NEXT VALUE FOR order_seq
  WHERE custno = 123456;
```

o

```
VALUES NEXT VALUE FOR order_seq INTO :hv_seq;
```

Información relacionada:

- “Identificadores” en la página 64
- “Predicado TYPE” en la página 237
- “CHAR” en la página 304
- “INTEGER” en la página 379
- “Selección completa” en la página 508
- “Sentencia CREATE TABLE” en la publicación *Consulta de SQL, Volumen 2*
- “Métodos” en la página 173
- “Sentencia CREATE FUNCTION (Escalar de SQL, tabla o fila)” en la publicación *Consulta de SQL, Volumen 2*
- “Conversiones entre tipos de datos” en la página 107
- “Asignaciones y comparaciones” en la página 110
- “Reglas para los tipos de datos del resultado” en la página 125
- “Reglas para la conversión de series” en la página 129

Predicados

Predicados

Predicados

Un *predicado* especifica una condición que es cierta, falsa o desconocida acerca de una fila o un grupo determinado.

Las siguientes reglas se aplican a todos los tipos de predicados:

- Todos los valores especificados en un predicado debe ser compatibles.
- Una expresión que se utiliza en un predicado básico, cuantificado, IN o BETWEEN no debe dar como resultado una serie de caracteres con un atributo de longitud superior a 4 000, una serie de caracteres gráficos con un atributo de longitud superior a 2 000 ni una serie LOB de cualquier tamaño.
- El valor de una variable del lenguaje principal puede ser nulo (es decir, la variable puede tener una variable indicadora negativa).
- La conversión de la página de códigos de los operandos de los predicados que implican dos o más operandos, a excepción de LIKE, se realiza según las reglas de conversión de series.
- La utilización de un valor DATALINK se limita al predicado NULL.
- La utilización de un valor de tipo estructurado está limitado al predicado NULL y al predicado TYPE.
- En una base de datos Unicode, todos los predicados que acepten una serie de caracteres o gráfica aceptarán todo tipo de serie para el que se soporte la conversión.

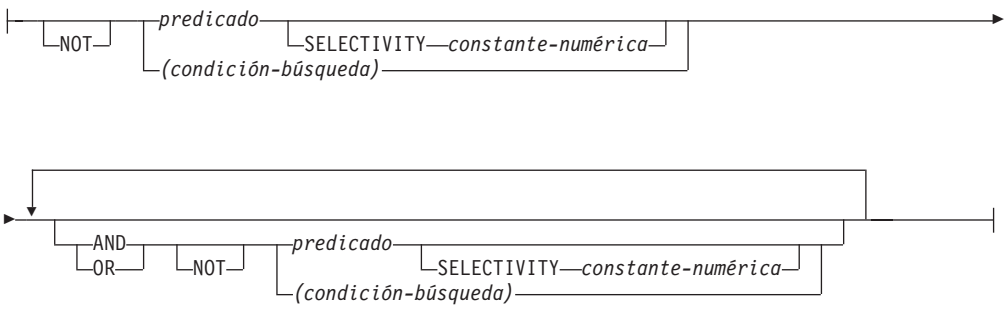
Una selección completa es una forma de sentencia SELECT que, cuando se utiliza en un predicado, también se denomina una *subconsulta*.

Información relacionada:

- “Selección completa” en la página 508
- “Reglas para la conversión de series” en la página 129

Condiciones de búsqueda

condición-búsqueda:



Una *condición de búsqueda* especifica una condición que es "verdadera," "falsa," o "desconocida" acerca de una fila determinada.

El resultado de una condición de búsqueda se deriva por la aplicación de *operadores lógicos* (AND, OR, NOT) especificados al resultado de cada predicado especificado. Si no se especifican operadores lógicos, el resultado de la condición de búsqueda es el resultado del predicado especificado.

AND y OR se definen en la Tabla 14, en la que P y Q son unos predicados cualesquiera:

Tabla 14. Tablas de evaluación para AND y OR

P	Q	P AND Q	P OR Q
Verdadero	Verdadero	Verdadero	Verdadero
Verdadero	Falso	Falso	Verdadero
Verdadero	Desconocido	Desconocido	Verdadero
Falso	Verdadero	Falso	Verdadero
Falso	Falso	Falso	Falso
Falso	Desconocido	Falso	Desconocido
Desconocido	Verdadero	Desconocido	Verdadero
Desconocido	Falso	Falso	Desconocido
Desconocido	Desconocido	Desconocido	Desconocido

NOT(verdadero) es falso, NOT(falso) es verdadero y NOT(desconocido) es desconocido.

En primer lugar se evalúan las condiciones de búsqueda entre paréntesis. Si el orden de evaluación no se especifica mediante paréntesis, NOT se aplica antes que AND y AND es aplica antes que OR. El orden en el que se evalúan los operadores del mismo nivel de prioridad no está definido, para permitir la optimización de condiciones de búsqueda.

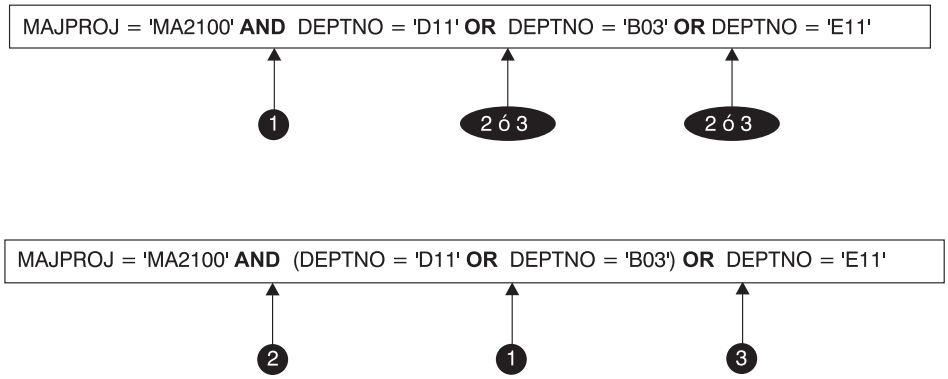


Figura 12. Orden de evaluación de las condiciones de búsqueda

SELECTIVITY *valor*

La cláusula SELECTIVITY se utiliza para indicar a DB2 qué porcentaje de selectividad prevista corresponde al predicado. SELECTIVITY se puede especificar sólo cuando el predicado es un predicado definido por el usuario.

Un predicado definido por el usuario consta de una invocación de función definida por el usuario, en el contexto de una especificación de predicado que coincide con la existente en la cláusula PREDICATES de CREATE FUNCTION. Por ejemplo, si la función `foo` está definida con PREDICATES WHEN=1..., es válido utilizar SELECTIVITY de este modo:

```
SELECT *
       FROM STORES
       WHERE foo(parm,parm) = 1 SELECTIVITY 0.004
```

El valor de selectividad debe ser un valor literal numérico comprendido dentro del rango inclusivo 0-1 (SQLSTATE 42615). Si SELECTIVITY no se especifica, el valor por omisión es 0.01 (es decir, el predicado definido por el usuario debe descartar todas las filas de la tabla excepto un 1 por ciento. El valor por omisión de SELECTIVITY se puede modificar para una función determinada actualizando su columna SELECTIVITY en la vista SYSSTAT.ROUTINES. Se obtiene un error si la cláusula SELECTIVITY se especifica para un predicado no definido por el usuario (SQLSTATE 428E5).

Se puede utilizar una función definida por el usuario (UDF) como predicado definido por el usuario y, por tanto, puede permitir la utilización de índices si:

- La especificación de predicado está presente en la sentencia CREATE FUNCTION
- la UDF se invoca en una cláusula WHERE que se compara (sintácticamente) de la misma manera que se especifica en la especificación de predicado
- no existe ninguna negación (operador NOT)

Ejemplos:

En la consulta siguiente, la especificación UDF `interna` de la cláusula WHERE cumple las tres condiciones y se considera que es un predicado definido por el usuario.

```
SELECT *
     FROM customers
     WHERE within(location, :sanJose) = 1 SELECTIVITY  0.2
```

Sin embargo, la presencia de `within` en la consulta siguiente no permite el uso de índices debido a la negación, y no se considera un predicado definido por el usuario.

```
SELECT *
     FROM customers
     WHERE NOT(within(location, :sanJose) = 1) SELECTIVITY  0.3
```

En el ejemplo siguiente, se identifican los clientes y tiendas que están a una determinada distancia entre sí. La distancia de una tienda a otra se calcula mediante el radio de la ciudad donde viven los clientes.

```
SELECT *
     FROM customers, stores
     WHERE distance(customers.loc, stores.loc) <
        CityRadius(stores.loc) SELECTIVITY 0.02
```

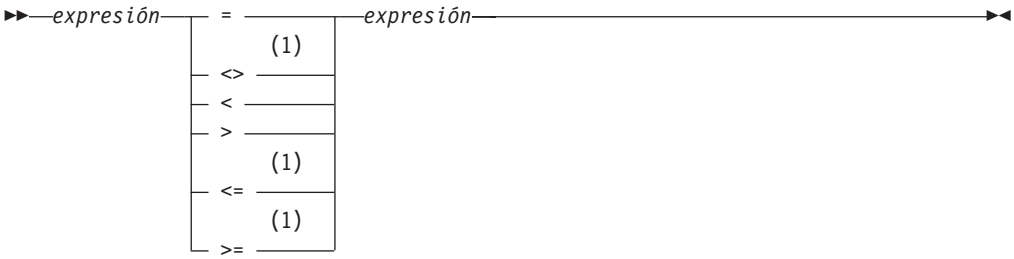
En la consulta anterior, se considera que el predicado contenido en la cláusula WHERE es un predicado definido por el usuario. El resultado producido por CityRadius se utiliza como argumento de búsqueda para la función productora de rangos.

Sin embargo, como el resultado devuelto por CityRadius se utiliza como función productora de rangos, el predicado definido por el usuario no podrá utilizar la extensión de índice definida para la columna stores.loc. Por lo tanto, la UDF sólo utilizará el índice definido en la columna customers.loc.

Información relacionada:

- "Sentencia CREATE FUNCTION (Escalar externa)" en la publicación *Consulta de SQL, Volumen 2*

Predicado básico



Notas:

1 También se da soporte a los formatos siguientes de operadores de comparación en predicados básicos y cuantificados: ^=, ^<, ^>, !=, !< y !>. En las páginas de códigos 437, 819 y 850, se da soporte a los formatos ¬=, ¬< y ¬>.

Todos estos formatos específicos para el producto de los operadores de comparación solamente están pensados para dar soporte al SQL existente que emplea estos operadores y no se recomienda su utilización para escribir sentencias de SQL nuevas.

Un *predicado básico* compara dos valores.

Si el valor de cualquier operando es nulo, el resultado del predicado será desconocido. De lo contrario el resultado es verdadero o falso.

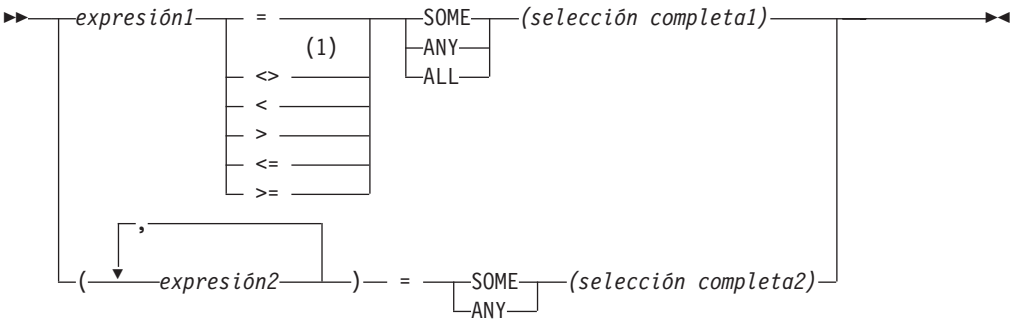
Para valores *x* e *y*:

Predicado	**Es verdadero si y sólo si...**
x **=** *y*	*x* es igual a *y*
x **<>** *y*	*x* es diferente de *y*
x **<** *y*	*x* es menor que *y*
x **>** *y*	*x* es mayor que *y*
x **>=** *y*	*x* es mayor o igual que *y*
x **<=** *y*	*x* es menor o igual que *y*

Ejemplos:

```
   EMPNO='528671'
SALARY < 20000
PRSTAFF <> :VAR1
SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE)
```

Predicado cuantificado



Notas:

1 También se da soporte a los formatos siguientes de operadores de comparación en predicados básicos y cuantificados: ^=, ^<, ^>, !=, !< y !>. En las páginas de códigos 437, 819 y 850, se da soporte a los formatos ¬=, ¬< y ¬>.

Todos estos formatos específicos para el producto de los operadores de comparación solamente están pensados para dar soporte al SQL existente que emplea estos operadores y no se recomienda su utilización para escribir sentencias de SQL nuevas.

Un *predicado cuantificado* compara un valor o valores con un grupo de valores.

La selección completa debe identificar un número de columnas que sea el mismo que el número de expresiones especificadas a la izquierda del operador del predicado (SQLSTATE 428C4). La selección completa puede devolver cualquier número de filas.

Cuando se especifica ALL:

- El resultado del predicado es verdadero si la selección completa no devuelve ningún valor o si la relación especificada es verdadera para cada valor que devuelva la selección completa.
- El resultado es falso si la relación especificada es falsa para un valor como mínimo que devuelve la selección completa.
- El resultado es desconocido si la relación especificada no es falsa para ninguno de los valores que devuelve la selección completa y una comparación como mínimo es desconocida debido a un valor nulo.

Cuando se especifica SOME o ANY:

- El resultado del predicado es verdadero si la relación especificada es verdadera para cada valor de una fila como mínimo que devuelve la selección completa.
- El resultado es falso si la selección completa no devuelve ninguna fila o si la relación especificada es falsa para como mínimo un valor de cada fila que devuelve la selección completa.
- El resultado es desconocido si la relación especificada no es verdadera para cualquiera de las filas y, como mínimo, una comparación es desconocida debido a un valor nulo.

Ejemplos: Utilice las tablas siguientes al hacer referencia a los ejemplos siguientes.

TBL*AB*:

COL*A*	COL*B*
1	12
2	12
3	13
4	14
-	-

TBL*XY*:

COL*X*	COL*Y*
2	22
3	23

Figura 13.

Ejemplo 1

```
SELECT COLA FROM TBLAB
   WHERE COLA = ANY(SELECT COLX FROM TBLXY)
```

Da como resultado 2,3. La subselección devuelve (2,3). COLA en las filas 2 y 3 es igual al menos a uno de estos valores.

Ejemplo 2

```
SELECT COLA FROM TBLAB
   WHERE COLA > ANY(SELECT COLX FROM TBLXY)
```

Da como resultado 3,4. La subselección devuelve (2,3). COLA en las filas 3 y 4 es mayor que al menos uno de estos valores.

Ejemplo 3

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY)
```

Da como resultado 4. La subselección devuelve (2,3). COLA en la fila 4 es el único que es mayor que estos dos valores.

Ejemplo 4

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY
                    WHERE COLX<0)
```

Da como resultado 1,2,3,4, nulo. La subselección no devuelve ningún valor. Por lo tanto, el predicado es verdadero para todas las filas de TBLAB.

Ejemplo 5

```
SELECT * FROM TBLAB
   WHERE (COLA,COLB+10) = SOME (SELECT COLX, COLY FROM TBLXY)
```

La subselección devuelve todas las entradas de TBLXY. El predicado es verdadero para la subselección, por lo tanto el resultado es el siguiente:

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

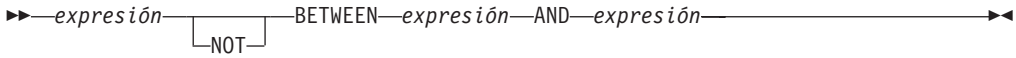
Ejemplo 6

```
SELECT * FROM TBLAB
   WHERE (COLA,COLB) = ANY (SELECT COLX,COLY-10 FROM TBLXY)
```

La subselección devuelve COLX y COLY-10 de TBLXY. El predicado es verdadero para la subselección, por lo tanto el resultado es el siguiente:

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

Predicado BETWEEN



El predicado BETWEEN compara un valor con un rango de valores.

El predicado BETWEEN:

```
valor1 BETWEEN valor2 AND valor3
```

es equivalente a la condición de búsqueda:

```
valor1 >= valor2 AND valor1 <= valor3
```

El predicado BETWEEN:

```
valor1 NOT BETWEEN valor2 AND valor3
```

es equivalente a la condición de búsqueda:

```
NOT(valor1 BETWEEN valor2 AND valor3); es decir,
valor1 < valor2 OR valor1 > valor3.
```

El primer operando (expresión) no puede incluir ninguna función que sea variante o que tenga una acción externa (SQLSTATE 426804).

En una mezcla de valores de indicación de fecha y hora y representaciones de serie de caracteres, todos los valores se convierten al tipo de datos del operando de fecha y hora.

Ejemplos:

Ejemplo 1

```
EMPLOYEE.SALARY BETWEEN 20000 AND 40000
```

Devuelve todos los salarios comprendidos entre 20.000 y 40.000 dólares.

Ejemplo 2

```
SALARY NOT BETWEEN 20000 + :HV1 AND 40000
```

Suponiendo que :HV1 es 5000, da como resultado todos los salarios que son inferiores a 25.000 dólares y superiores a 40.000.

Predicado EXISTS

El predicado EXISTS comprueba la existencia de ciertas filas.

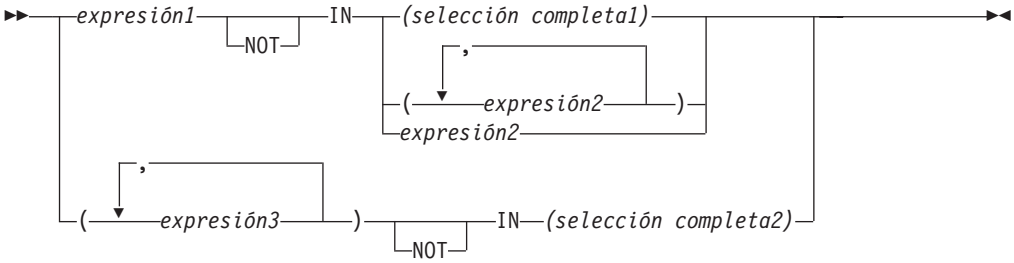
La selección completa puede especificar cualquier número de columnas y

- El resultado es verdadero sólo si el número de filas especificadas mediante la selección completa no es cero.
- El resultado es falso sólo si el número de filas especificadas es cero
- El resultado no puede ser desconocido.

Ejemplo:

```
EXISTS (SELECT * FROM TEMPL WHERE SALARY < 10000)
```

Predicado IN



El predicado IN compara un valor o valores con un conjunto de valores.

La selección completa debe identificar un número de columnas que sea el mismo que el número de expresiones especificadas a la izquierda de la palabra clave IN (SQLSTATE 428C4). La selección completa puede devolver cualquier número de filas.

- Un predicado IN del formato:

  ```
  expresión IN expresión
  ```

  es equivalente a un predicado básico del formato:

  ```
  expresión = expresión
  ```

- Un predicado IN del formato:

  ```
  expresión IN (selección completa)
  ```

  es equivalente a un predicado cuantificado del formato:

  ```
  expresión = ANY (selección completa)
  ```

- Un predicado IN del formato:

  ```
  expresión NOT IN (selección completa)
  ```

  es equivalente a un predicado cuantificado del formato:

  ```
  expresión <> ALL (selección completa)
  ```

- Un predicado IN del formato:

  ```
  expresión IN (expresióna, expresiónb, ..., expresiónk)
  ```

  es equivalente a:

  ```
  expresión = ANY (selección completa)
  ```

  donde selección completa en el formato de la cláusula-values es:

  ```
  VALUES (expresióna), (expresiónb), ..., (expresiónk)
  ```

- Un predicado IN del formato:

  ```
  (expresióna, expresiónb,..., expresiónk) IN (selección completa)
  ```

  es equivalente a un predicado cuantificado del formato:

  ```
  (expresióna, expresiónb,..., expresiónk) = ANY (selección completa)
  ```

Los valores para *expresión1* y *expresión2* o la columna de *selección completa1* del predicado IN deben ser compatibles. Cada valor de *expresión3* y su columna correspondiente de *selección completa2* del predicado IN deben ser compatibles. Pueden utilizarse las reglas para tipos de datos del resultado para determinar los atributos del resultado utilizados en la comparación.

Los valores para las expresiones del predicado IN (incluyendo las columnas correspondientes de una selección completa)pueden tener páginas de códigos diferentes. Si se precisa realizar una conversión, la página de códigos se determina aplicando las reglas para las conversiones de series a la lista IN primero y, posteriormente, al predicado, utilizando la página de códigos derivada para la lista IN como segundo operando.

Ejemplos:

Ejemplo 1: Lo siguiente es verdadero si el valor de la fila bajo evaluación de la columna DEPTNO contiene D01, B01 o C01:

```
DEPTNO IN ('D01', 'B01', 'C01')
```

Ejemplo 2: Lo siguiente se considera verdadero sólo si EMPNO (número de empleado) a la izquierda coincide con EMPNO de un empleado del departamento E11:

```
EMPNO IN (SELECT EMPNO FROM EMPLOYEE WHERE WORKDEPT = 'E11')
```

Ejemplo 3: Dada la siguiente información, este ejemplo se considera verdadero si el valor específico de la fila de la columna COL_1 coincide con cualquier valor de la lista:

Tabla 15. Ejemplo de predicado IN

Expresiones	Tipo	Página de códigos
COL_1	columna	850
HV_2	variable del lenguaje principal	437
HV_3	variable del lenguaje principal	437
CON_1	constante	850

Cuando se evalúa el predicado:

```
COL_1 IN (:HV_2, :HV_3, CON_4)
```

las dos variables del lenguaje principal se convertirán a la página de códigos 850, en base a las reglas para las conversiones de series.

Ejemplo 4: Lo siguiente se considera verdadero si el año especificado en EMENDATE (la fecha en que ha finalizado la actividad de un empleado en un proyecto) coincide con cualquiera de los valores especificados en la lista (el año actual o los dos años anteriores):

```
YEAR(EMENDATE) IN (YEAR(CURRENT DATE),
                   YEAR(CURRENT DATE - 1 YEAR),
                   YEAR(CURRENT DATE - 2 YEARS))
```

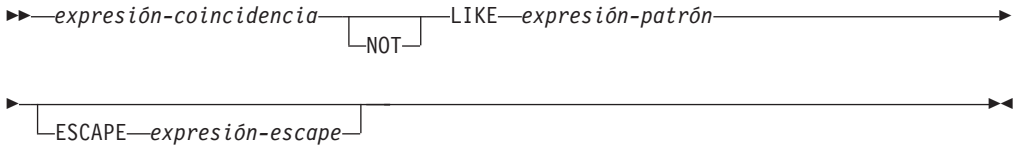
Ejemplo 5: Lo siguiente se considera verdadero si tanto ID como DEPT del lado izquierdo coinciden con MANAGER y DEPTNUMB respectivamente para cualquier fila de la tabla ORG.

```
(ID, DEPT) IN (SELECT MANAGER, DEPTNUMB FROM ORG)
```

Información relacionada:

- "Reglas para los tipos de datos del resultado" en la página 125
- "Reglas para la conversión de series" en la página 129

Predicado LIKE



El predicado LIKE busca series que tienen un determinado patrón. El patrón se especifica mediante una serie en la que el signo de subrayado y de porcentaje pueden tener un significado especial. Los blancos finales de un patrón forman parte del mismo.

Si el valor de cualquiera de los argumentos es nulo, el resultado del predicado LIKE es desconocido.

Los valores de *expresión-coincidencia*, *expresión-patrón* y *expresión-escape* son expresiones de serie compatibles. Hay ligeras diferencias en los tipos de expresiones de series soportados por cada uno de los argumentos. Los tipos válidos de las expresiones se listan bajo la descripción de cada argumento.

Ninguna de las expresiones puede tener un tipo diferenciado. Sin embargo, pueden ser una función que convierta un tipo diferenciado en su tipo fuente.

expresión-coincidencia
Una expresión que especifica la serie que se debe examinar para ver si cumple con un determinado patrón de caracteres.

La expresión se puede especificar mediante:

- Una constante
- Un registro especial
- Una variable del lenguaje principal (incluida una variable localizadora o una variable de referencia de archivo)
- Una función escalar
- Un localizador de objeto grande
- Un nombre de columna
- Una expresión que concatene cualquiera de las anteriores

expresión-patrón
Una expresión que especifica la serie que se debe comparar.

La expresión se puede especificar mediante:

- Una constante
- Un registro especial
- Una variable del lenguaje principal
- Una función escalar cuyos operandos seas cualquiera de los anteriores
- Una expresión que concatene cualquiera de las anteriores

con las siguientes restricciones:

- Ningún elemento de la expresión puede ser del tipo LONG VARCHAR, CLOB, LONG VARGRAPHIC o DBCLOB. Además, no puede tratarse de una variable de referencia de archivo BLOB.
- La longitud real de *expresión-patrón* no puede superar los 32.672 bytes.

Una **descripción sencilla** del uso del predicado LIKE es que el patrón se utilice para especificar los criterios de cumplimiento correspondientes a los valores de *expresión-coincidencia*, donde:

- El carácter de subrayado (_) representa cualquier carácter único.
- El signo de porcentaje (%) representa una serie de cero o más caracteres.
- Cualquier otro carácter se representa a sí mismo.

Si la *expresión-patrón* tiene que incluir el carácter de subrayado o de porcentaje, la *expresión-escape* se utiliza para especificar un carácter que precede al carácter de subrayado o de porcentaje en el patrón.

A continuación se ofrece una **descripción rigurosa** del predicado LIKE. Tenga en cuenta que en esta descripción se pasa por alto el uso de la *expresión-escape*; su uso se explicará más adelante.

- Supongamos que *m* indica el valor de *expresión-coincidencia* y que *p* indica el valor de *expresión-patrón*. La serie *p* se interpreta como una secuencia el número mínimo de especificadores de subserie, de modo que cada carácter de *p* forma parte de exactamente un especificador de subserie. Un especificador de subserie es un carácter de subrayado, un signo de porcentaje o una secuencia no vacía de caracteres que no son el signo de subrayado ni de porcentaje.

  El resultado del predicado es desconocido si *m* o *p* es el valor nulo. De lo contrario, el resultado es verdadero (true) o falso (false). El resultado es true si *m* y *p* son ambos series vacías o existe un particionamiento de *m* en subseries como:

  – Una subserie de *m* es una secuencia de cero o más caracteres contiguos y cada carácter de *m* forma parte de exactamente una subserie.
  – Si el especificador de subserie número *n* es un carácter de subrayado, la subserie número *n* de *m* es cualquier carácter único.
  – Si el especificador de subserie número *n* es un carácter de porcentaje, la subserie número *n* de *m* es cualquier secuencia de cero o más caracteres.
  – Si el especificador de subserie número *n* no es ni un signo de subrayado ni uno de porcentaje, la subserie número *n* de *m* es igual a dicho especificador de subserie y tiene la misma longitud que dicho especificador de subserie.
  – El número de subseries de *m* es igual al número de especificadores de subserie.

  Por lo tanto, si *p* es una serie vacía y *m* no es una serie vacía, el resultado es false. Paralelamente, si *m* es una serie vacía y *p* no es una serie vacía (excepto para una serie que solo contenga signos de porcentaje), el resultado es false.

  El predicado *m* NOT LIKE *p* es equivalente a la condición de búsqueda NOT (*m* LIKE *p*).

Cuando se especifica la *expresión-escape*, la *expresión-patrón* no debe contener el carácter de escape especificado por la *expresión-escape*, excepto cuando va seguido inmediatamente del carácter de escape, el carácter de subrayado o el carácter de porcentaje (SQLSTATE 22025).

Si la *expresión-coincidencia* es una serie de caracteres en una base de datos MBCS, puede contener datos mixtos. En este caso, el patrón puede incluir tanto caracteres SBCS como no SBCS. Para bases de datos que no son Unicode, los caracteres especiales del patrón se interpretan del siguiente modo:

- Un signo de subrayado de media anchura SBCS hace referencia a un carácter SBCS.
- Un carácter de subrayado de anchura completa no SBCS hace referencia a un carácter no SBCS.
- Un signo de porcentaje de media anchura SBCS o de anchura completa no SBCS hace referencia a cero o más caracteres SBCS o no SBCS.

En una base de datos Unicode, no se suele hacer distinción entre caracteres de ″un solo byte″ y ″no de un solo byte″. Aunque el formato UTF-8 es una codificación de ″bytes mixtos″ de caracteres Unicode, no existe ninguna distinción real entre caracteres SBCS y no SBCS en UTF-8. Cada carácter es un carácter Unicode, independientemente del número de bytes en formato UTF-8.

En una columna gráfica Unicode, cada carácter que no sea suplementario, incluido el carácter de signo de subrayado de media anchura (U+005F) y el carácter de signo de porcentaje de media anchura (U+0025), tiene dos bytes de anchura. En una base de datos Unicode, los caracteres especiales de un patrón se interpretan del siguiente modo:

- Para series de caracteres, un carácter de subrayado de media anchura (X'5F') o un carácter de subrayado de anchura completa (X'EFBCBF') hace referencia a un carácter Unicode y un carácter de signo de porcentaje de media anchura (X'25') o un carácter de signo de porcentaje de anchura completa (X'EFBC85') hace referencia a cero o más caracteres Unicode.
- Para series gráficas, un carácter de subrayado de media anchura (U+005F) o un carácter de subrayado de anchura completa (U+FF3F) hace referencia a un carácter Unicode y un carácter de signo de porcentaje de media anchura (U+0025) o un carácter de signo de porcentaje de anchura completa (U+FF05) hace referencia a cero o más caracteres Unicode.

Necesita dos caracteres de subrayado para comparar un carácter gráfico suplementario Unicode, puesto que dicho carácter se representa mediante dos caracteres UCS-2 en una columna gráfica. Sin embargo, sólo necesita un carácter de subrayado para comparar un carácter suplementario Unicode en una columna de carácter.

expresión-escape
Este argumento opcional es una expresión que especifica un carácter que se utilizará para modificar el significado especial de los caracteres de subrayado (_) y de porcentaje (%) en la *expresión-patrón*. Esto permite utilizar el predicado LIKE para comparar valores que contienen los caracteres reales de porcentaje y de subrayado.

La expresión se puede especificar mediante:

- una constante
- un registro especial
- una variable del lenguaje principal
- una función escalar cuyos operandos son cualquiera de los mencionados anteriormente
- una expresión que concatene cualquiera de los elementos anteriores

teniendo en cuenta las siguientes restricciones:

- Ningún elemento de la expresión puede ser del tipo LONG VARCHAR, CLOB, LONG VARGRAPHIC o DBCLOB. Además, no puede ser una variable de referencia de archivo BLOB.

- Para columnas de caracteres, el resultado de la expresión debe ser un carácter o una serie binaria que contenga exactamente un byte (SQLSTATE 22019).
- Para columnas gráficas, el resultado de la expresión debe ser un carácter (SQLSTATE 22019).

Cuando hay caracteres de escape en la serie de patrón, un carácter de subrayado, de porcentaje o de escape puede representar una ocurrencia literal de sí mismo. Esto es cierto si el carácter en cuestión va precedido de un número impar de caracteres de escape sucesivos. De lo contrario, no es cierto.

En un patrón, una secuencia de caracteres de escape sucesivos se trata del siguiente modo:

- Supongamos que S es una secuencia y que no forma parte de una secuencia más larga de caracteres de escape sucesivos. Supongamos también que S contiene un total de n caracteres. Las reglas que controlan S dependen del valor de n:
  - Si n es impar, S debe ir seguido de un signo de subrayado o de porcentaje (SQLSTATE 22025). S y el carácter que lo sigue representan (n-1)/2 ocurrencias literales del carácter de escape seguidas de una ocurrencia literal del signo de subrayado o de porcentaje.
  - Si n es par, S representa n/2 ocurrencias literales del carácter de escape. A diferencia del caso e que n es impar, S puede finalizar el patrón. Si no finaliza el patrón, puede ir seguido de cualquier carácter (excepto, por supuesto, de un carácter de escape, que violaría la suposición de que S no forma parte de una secuencia más larga de caracteres de escape sucesivos). Si S va seguido de un signo de subrayado o de porcentaje, dicho carácter tiene su significado especial.

A continuación se ilustra el efecto de ocurrencias sucesivas del carácter de escape que, en este caso, es la barra inclinada invertida (\).

Serie de patrón	**Patrón real**
\%	Un signo de porcentaje
\\%	Una barra inclinada invertida seguida de cero o más caracteres arbitrarios
\\\%	Una barra inclinada invertida seguida de un signo de porcentaje

La página de códigos utilizada en la comparación se basa en la página de códigos del valor de *expresión-coincidencia*.

- El valor de *expresión-coincidencia* nunca se convierte.
- Si la página de códigos de *expresión-patrón* difiere de la página de códigos de *expresión-coincidencia*, el valor de *expresión-patrón* se convierte a la página de códigos de *expresión-coincidencia*, a no ser que alguno de los operandos esté definido como FOR BIT DATA (en cuyo caso no hay conversión).
- Si la página de códigos de *expresión-escape* difiere de la página de códigos de *expresión-coincidencia*, el valor de *expresión-escape* se convierte a la página de códigos de *expresión-coincidencia*, a no ser que alguno de los operandos esté definido como FOR BIT DATA (en cuyo caso no hay conversión).

Notas:

- El número de blancos finales es significativo tanto en *expresión-coincidencia* como en *expresión-patrón*. Si las series no tienen la misma longitud, la serie más corta no se rellena con espacios en blanco. Por ejemplo, la expresión `'PADDED ' LIKE 'PADDED'` no daría lugar a una coincidencia.
- Si el patrón especificado en un predicado LIKE es un marcador de parámetro y se utiliza una variable del lenguaje principal de caracteres de longitud fija para sustituir el marcador de parámetro, el valor especificado para la variable del lenguaje principal debe tener la longitud correcta. Si no se especifica la longitud correcta, la operación select no devolverá los resultados previstos.

  Por ejemplo, si la variable del lenguaje principal se define como CHAR(10) y se asigna el valor WYSE% a dicha variable del lenguaje principal, la variable del lenguaje principal se rellena con blancos durante la asignación. El patrón utilizado es:

  ```
  'WYSE%     '
  ```

  El gestor de bases de datos busca todos los valores que comienzan por WYSE y terminan por cinco espacios en blanco. Si desea buscar sólo valores que comienzan por 'WYSE', asigne el valor 'WSYE%%%%%%' a la variable del lenguaje principal.

Ejemplos:

- Busque la serie 'SYSTEMS' que aparezca en cualquier lugar dentro de la columna PROJNAME de la tabla PROJECT.

  ```
  SELECT PROJNAME FROM PROJECT
    WHERE PROJECT.PROJNAME LIKE '%SYSTEMS%'
  ```

- Busque una serie cuyo primer carácter sea 'J' que tenga exactamente dos caracteres de longitud en la columna FIRSTNME de la tabla EMPLOYEE.

  ```
  SELECT FIRSTNME FROM EMPLOYEE
    WHERE EMPLOYEE.FIRSTNME LIKE 'J_'
  ```

- Busque una serie de cualquier longitud, cuyo primer carácter sea 'J', en la columna FIRSTNME de la tabla EMPLOYEE.

  ```
  SELECT FIRSTNME FROM EMPLOYEE
    WHERE EMPLOYEE.FIRSTNME LIKE 'J%'
  ```

- En la tabla CORP_SERVERS, busque una serie en la columna LA_SERVERS que coincida con el valor del registro especial CURRENT SERVER.

  ```
  SELECT LA_SERVERS FROM CORP_SERVERS
    WHERE CORP_SERVERS.LA_SERVERS LIKE CURRENT SERVER
  ```

- Recupere todas las series que comienzan por la secuencia de caracteres '%_\' en la columna A de la tabla T.

  ```
  SELECT A FROM T
    WHERE T.A LIKE '\%\_\\%' ESCAPE '\'
  ```

- Utilice la función escalar BLOB para obtener un carácter de escape de un byte que sea compatible con los tipos de datos de patrón y de coincidencia (ambos BLOB).

  ```
  SELECT COLBLOB FROM TABLET
    WHERE COLBLOB LIKE :var_patrón ESCAPE BLOB(X'0E')
  ```

Predicado NULL

El predicado NULL comprueba la existencia de valores nulos.

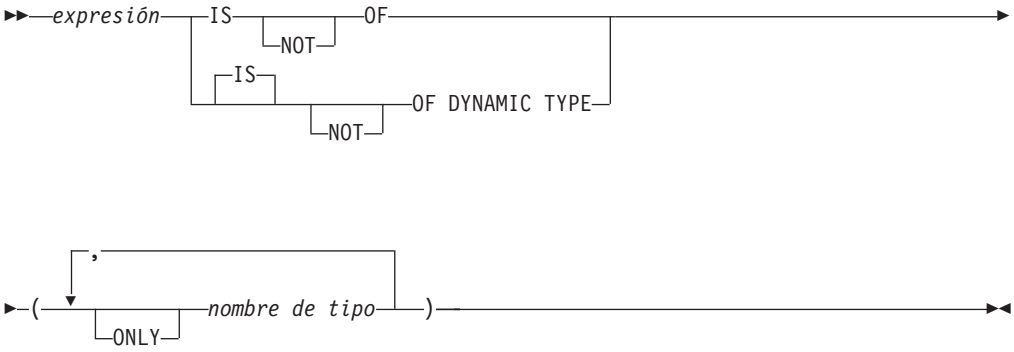
El resultado de un predicado NULL no puede ser desconocido. Si el valor de la expresión es nulo, el resultado es verdadero. Si el valor no es nulo, el resultado es falso. Si se especifica NOT, el resultado se invierte.

Ejemplos:

```
PHONENO IS NULL

SALARY IS NOT NULL
```

Predicado TYPE



Un *predicado TYPE* compara el tipo de una expresión con uno o más tipos estructurados definidos por el usuario.

El tipo dinámico de una expresión que implica desreferenciar un tipo de referencia es el tipo real de la fila referenciada de la tabla o vista con tipo de destino. Puede diferenciarse del tipo de destino de una expresión que implica la referencia, denominado el tipo estático de la expresión.

Si el valor de *expresión* es nulo, el resultado del predicado será desconocido. El resultado del predicado será verdadero si el tipo dinámico de la *expresión* es un subtipo de uno de los tipos estructurados especificados por *nombre de tipo*; de lo contrario, el resultado será falso. Si ONLY precede cualquier *nombre de tipo*, no se tienen en cuenta los subtipos correspondientes de este tipo.

Si *nombre de tipo* no está calificado, se resuelve utilizando la vía de acceso de SQL. Cada *nombre de tipo* debe identificar un tipo definido por el usuario que esté en la jerarquía de tipos del tipo estático de *expresión* (SQLSTATE 428DU).

Debe utilizarse la función DEREF siempre que el predicado TYPE tenga una expresión que implique un valor de tipo de referencia. El tipo estático de esta forma de *expresión* es el tipo de destino de la referencia.

La sintaxis IS OF y OF DYNAMIC TYPE son alternativas equivalentes para el predicado TYPE. Asimismo, IS NOT OF y NOT OF DYNAMIC TYPE son alternativas equivalentes.

Ejemplos:

Existe una jerarquía de tablas que tiene una tabla raíz EMPLOYEE de tipo EMP y una subtabla MANAGER de tipo MGR. Otra tabla, ACTIVITIES, incluye una columna denominada WHO_RESPONSIBLE que está definida como REF(EMP) SCOPE EMPLOYEE. Lo siguiente es un predicado de tipo que devuelve un resultado verdadero cuando una fila correspondiente a WHO_RESPONSIBLE es un director (″manager″):

```
DEREF (WHO_RESPONSIBLE) IS OF (MGR)
```

Si una tabla contiene una columna EMPLOYEE de tipo EMP, EMPLOYEE puede contener valores de tipo EMP y también valores de sus subtipos, tales como MGR. El predicado siguiente

```
EMPL IS OF (MGR)
```

devuelve un resultado verdadero cuando EMPL no es nulo y es realmente un director.

Información relacionada:

- "DEREF" en la página 333

Capítulo 3. Funciones

Resumen de las funciones

Una *función* es una operación que se indica mediante un nombre de función seguido por un par de paréntesis que contienen la especificación de los argumentos (es posible que no haya argumentos).

Las *funciones incorporadas* las proporciona el gestor de bases de datos; devuelven un resultado de un solo valor y se identifican como parte del esquema SYSIBM. Entre las funciones incorporadas se incluyen las funciones de columna (como, por ejemplo, AVG), las funciones con operadores (por ejemplo, "+"), las funciones de conversión (como DECIMAL), y otras (como SUBSTR).

Las *funciones definidas por el usuario* se registran en una base de datos de SYSCAT.ROUTINES (utilizando la sentencia CREATE FUNCTION). Estas funciones nunca forman parte del esquema SYSIBM. Se proporciona un conjunto de estas funciones con el gestor de bases de datos en un esquema denominado SYSFUN y otro en un esquema denominado SYSPROC.

Las funciones se clasifican como funciones agregadas (de columna), funciones escalares, funciones de fila y funciones de tabla.

- El argumento de una *función de columna* es un conjunto de valores similares. Una función de columna devuelve un solo valor (posiblemente nulo) y puede especificarse en una sentencia de SQL donde sea posible utilizar una expresión.
- Los argumentos de una *función escalar* son valores escalares individuales, que pueden ser de tipos distintos y tener significados diferentes. Una función escalar devuelve un solo valor (posiblemente nulo) y puede especificarse en una sentencia de SQL donde sea posible utilizar una expresión.
- El argumento de una *función de fila* es un tipo estructurado. Una función de fila devuelve una fila de tipos de datos incorporados y sólo se puede especificar como función de transformación para un tipo estructurado.
- Los argumentos de una *función de tabla* son valores escalares individuales, que pueden ser de tipos distintos y tener significados diferentes. Una función de tabla devuelve una tabla a la sentencia SQL y sólo puede especificarse en la cláusula FROM de una sentencia SELECT.

El nombre de la función, combinado con el esquema, proporciona el nombre completamente calificado de la función. La combinación del esquema, el nombre de función y los parámetros de entrada constituye una *signatura de función*.

En algunos casos, el tipo de parámetro de entrada se especifica como un tipo de datos incorporados concreto y, en otros casos, se especifica mediante una variable general como *cualquier-tipo-numérico*. Si se especifica un tipo de datos concreto, una coincidencia exacta sólo se obtendrá con el tipo de datos especificado. Si se utiliza una variable general, cada uno de los tipos de datos asociados con dicha variable da como resultado una coincidencia exacta.

Es posible que existan funciones adicionales, porque las funciones definidas por el usuario pueden crearse en esquemas distintos, utilizando como fuente una de las signaturas de función. También es posible crear funciones externas en las aplicaciones.

Conceptos relacionados:

- "Funciones agregadas" en la página 273

Información relacionada:

- "Funciones" en la página 164
- "Subselección" en la página 470
- "Sentencia CREATE FUNCTION" en la publicación *Consulta de SQL, Volumen 2*

Funciones soportadas y rutinas de administración de SQL

La Tabla 16 resume información sobre las funciones a las que se proporciona soporte y las rutinas de administración de SQL. El nombre de la función, junto con el esquema, ofrece el nombre completamente calificado de una función. La columna "Parámetros de entrada" muestra el tipo de datos esperado para cada argumento durante la invocación de la función. Muchas de las funciones incluyen variaciones de parámetros de entrada, lo que permite utilizar distintos tipos de datos o un número diferente de argumentos. La combinación de esquema, nombre de función y parámetros de entrada forma una signatura de función. La columna "Devuelve" muestra los tipos de datos posibles de los valores que devuelve la función.

Para ver las listas de las funciones incorporadas a las que se proporciona soporte clasificadas por tipos, consulte las tablas siguientes:

- Funciones agregadas (Tabla 17 en la página 265)
- Funciones escalares de difusión (Tabla 18 en la página 266)
- Funciones escalares de enlace de datos (Tabla 19 en la página 267)
- Funciones escalares de fecha y hora (Tabla 20 en la página 267)
- Funciones escalares de particionamiento (Tabla 21 en la página 269)
- Funciones escalares numéricas (Tabla 22 en la página 269)
- Funciones escalares de series (Tabla 23 en la página 270)
- Funciones escalares diversas (Tabla 24 en la página 271)

Tabla 16. Funciones soportadas

Nombre de la función	Esquema / Parámetros de entrada	Descripción / Devuelve
"ABS o ABSVAL" en la página 293	SYSIBM	Esta función escalar devuelve el valor absoluto del argumento.
	Cualquier expresión que devuelva un tipo de datos numérico integrado.	Mismo tipo de datos y longitud que el argumento.
"ABS o ABSVAL" en la página 293	SYSFUN	Esta función escalar devuelve el valor absoluto del argumento.
	SMALLINT	SMALLINT
	INTEGER	INTEGER
	BIGINT	BIGINT
	DOUBLE	DOUBLE
"ACOS" en la página 294	SYSIBM	Esta función escalar devuelve el coarcocoseno del argumento como un ángulo expresado en radianes.
	DOUBLE	DOUBLE
ALTOBJ	SYSPROC	Este procedimiento (una rutina de administración de SQL) analiza una sentencia CREATE TABLE de entrada que sirve como el lenguaje de definición de datos (DDL) destino de una tabla existente que debe modificarse.
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.	
AM_BASE_RPT_RECOMS	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve recomendaciones para los informes de actividad que el supervisor de actividad utiliza.
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.	
AM_BASE_RPTS	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve los informes de actividad que el supervisor de actividad utiliza.
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.	

Funciones soportadas

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
AM_DROP_TASK	SYSPROC	Este procedimiento (una rutina de administración de SQL) suprime una tarea de supervisión.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
AM_GET_LOCK_CHN_TB	SYSPROC	Este procedimiento (una rutina de administración de SQL) devuelve una serie de bloqueo de aplicación en formato de tabla.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
AM_GET_LOCK_CHNS	SYSPROC	Este procedimiento (una rutina de administración de SQL) visualiza las series de bloqueo de una aplicación determinada utilizando una serie con formato.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
AM_GET_LOCK_RPT	SYSPROC	Este procedimiento (una rutina de administración de SQL) visualiza los detalles sobre el bloqueo de una aplicación.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
AM_GET_RPT	SYSPROC	Este procedimiento (una rutina de administración de SQL) visualiza los datos sobre el supervisor de actividad para un informe.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
AM_SAVE_TASK	SYSPROC	Este procedimiento (una rutina de administración de SQL) crea o modifica una tarea de supervisión.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
APP	DB2EAS	Este procedimiento (una rutina de administración de SQL) inicia o detiene una aplicación determinada en el servidor de aplicaciones de DB2.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
APPLICATION_ID	SYSFUN	Esta función escalar (una rutina de administración de SQL) devuelve el ID de aplicación de la conexión actual.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"ASCII" en la página 295	SYSFUN	Esta función escalar devuelve el valor de código ASCII del carácter situado más a la izquierda del argumento como un entero.	
	CHAR		INTEGER
	VARCHAR(4000)		INTEGER
	CLOB(1M)		INTEGER
"ASIN" en la página 296	SYSIBM	Esta función escalar devuelve el arcoseno del argumento como un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
"ATAN" en la página 297	SYSIBM	Esta función escalar devuelve la arcotangente del argumento como un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
"ATANH ATANH" en la página 299	SYSIBM	Esta función escalar devuelve la arcotangente hiperbólica del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
"ATAN2" en la página 298	SYSIBM	Esta función escalar devuelve la arcotangente de las coordenadas *x* e *y* (especificado por el primer y segundo argumentos respectivamente) como un ángulo, expresado en radianes.	
	DOUBLE, DOUBLE		DOUBLE

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción
	Parámetros de entrada	**Devuelve**
"AVG" en la página 274	SYSIBM	Esta función agregada devuelve el promedio de un conjunto de números.
	tipo-numérico[4]	*tipo-numérico*[1]
"BIGINT" en la página 300	SYSIBM	Esta función escalar devuelve una representación en el formato de un entero de 64 bits de un número o una serie de caracteres en el formato de una constante de enteros.
	tipo-numérico	BIGINT
	VARCHAR	BIGINT
"BLOB" en la página 302	SYSIBM	Esta función escalar convierte del tipo fuente a BLOB, con longitud opcional.
	tipo-serie	BLOB
	tipo-serie, INTEGER	BLOB
"CEILING o CEIL" en la página 303	SYSIBM	Esta función escalar devuelve el entero menor que es mayor o igual que el argumento.
	SMALLINT	SMALLINT
	INTEGER	INTEGER
	BIGINT	BIGINT
	DOUBLE	DOUBLE
"CHAR" en la página 304	SYSIBM	Esta función escalar devuelve una representación en el formato de una serie del tipo fuente.
	tipo-carácter	CHAR
	tipo-carácter, INTEGER	CHAR(*entero*)
	tipo-fechahora	CHAR
	tipo-fechahora, *palabra-clave*[2]	CHAR
	SMALLINT	CHAR(6)
	INTEGER	CHAR(11)
	BIGINT	CHAR(20)
	DECIMAL	CHAR(2+*precisión*)
	DECIMAL, VARCHAR	CHAR(2+*precisión*)
"CHAR" en la página 304	SYSFUN	Esta función escalar devuelve una representación en el formato de una serie de caracteres de un número de coma flotante.
	DOUBLE	CHAR(24)
"CHR" en la página 309	SYSFUN	Esta función escalar devuelve el carácter que tiene el valor del código ASCII especificado por el argumento. El valor del argumento debe estar comprendido entre 0 y 255; de lo contrario, el valor devuelto es nulo.
	INTEGER	CHAR(1)
"CLOB" en la página 310	SYSIBM	Esta función escalar convierte del tipo fuente a CLOB, con longitud opcional.
	tipo-carácter	CLOB
	tipo-carácter, INTEGER	CLOB
"COALESCE" en la página 311[3]	SYSIBM	Esta función escalar devuelve el primer argumento no nulo del conjunto de argumentos.
	cualquier-tipo, *cualquier-tipo-compatible-unión*, ...	*cualquier-tipo*
"CONCAT" en la página 312	SYSIBM	Esta función escalar devuelve la concatenación de dos argumentos de serie.
	tipo-serie, *tipo-serie-compatible*	*tipo-serie-máx*
"CORRELATION" en la página 276	SYSIBM	Esta función agregada devuelve el coeficiente de correlación de un conjunto de pares de números.
	tipo-numérico, *tipo-numérico*	DOUBLE

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"COS" en la página 313	SYSIBM	Esta función escalar devuelve el coseno del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
"COSH" en la página 314	SYSIBM	Esta función escalar devuelve el coseno hiperbólico del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
"COT" en la página 315	SYSIBM	Esta función escalar devuelve la cotangente del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
"COUNT" en la página 277	SYSIBM	Esta función agregada devuelve el número de filas o valores de un conjunto de filas o valores.	
	cualquier-tipo-integrado[4]		INTEGER
"COUNT_BIG" en la página 278	SYSIBM	Esta función agregada devuelve el número de filas o valores de un conjunto de filas o valores. El resultado puede ser mayor que el valor máximo de INTEGER.	
	cualquier-tipo-integrado[4]		DECIMAL(31,0)
"COVARIANCE" en la página 280	SYSIBM	Esta función agregada devuelve la covarianza de un conjunto de pares de números.	
	tipo-numérico, *tipo-numérico*		DOUBLE
"DATE" en la página 316	SYSIBM	Esta función escalar devuelve una fecha a partir de un solo valor de entrada.	
	DATE		DATE
	TIMESTAMP		DATE
	DOUBLE		DATE
	VARCHAR		DATE
"DAY" en la página 317	SYSIBM	Esta función escalar devuelve la parte correspondiente al día de un valor.	
	VARCHAR		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
"DAYNAME" en la página 318	SYSFUN	Esta función escalar devuelve una serie que combina caracteres en mayúsculas y minúsculas que contiene el nombre del día (por ejemplo, viernes) de la parte correspondiente al día del argumento, según el entorno local en el momento en que se emitió db2start.	
	VARCHAR(26)		VARCHAR(100)
	DATE		VARCHAR(100)
	TIMESTAMP		VARCHAR(100)
"DAYOFWEEK" en la página 319	SYSFUN	Esta función escalar devuelve el día de la semana del argumento como un valor entero comprendido entre 1 y 7, donde 1 representa el domingo.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
"DAYOFWEEK_ISO" en la página 320	SYSFUN	Esta función escalar devuelve el día de la semana del argumento como un valor entero comprendido entre 1 y 7, donde 1 representa el lunes.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"DAYOFYEAR" en la página 321	SYSFUN	Esta función escalar devuelve el día del año del argumento como un valor entero comprendido entre 1 y 366.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
"DAYS" en la página 322	SYSIBM	Esta función escalar devuelve una representación entera de una fecha.	
	VARCHAR		INTEGER
	TIMESTAMP		INTEGER
	DATE		INTEGER
DB_PARTITIONS	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve el contenido del archivo db2nodes.cfg en forma de tabla.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"DBCLOB" en la página 323	SYSIBM	Esta función escalar convierte del tipo fuente a DBCLOB, con longitud opcional.	
	tipo-gráfico		DBCLOB
	tipo-gráfico, INTEGER		DBCLOB
"DBPARTITIONNUM" en la página 324 [3]	SYSIBM	Esta función escalar devuelve el número de partición de base de datos de la fila. El argumento es un nombre de columna dentro de una tabla.	
	cualquier-tipo		INTEGER
"DECIMAL" en la página 326	SYSIBM	Esta función escalar devuelve una representación decimal de un número, con precisión y escala opcionales.	
	tipo-numérico		DECIMAL
	tipo-numérico, INTEGER		DECIMAL
	tipo-numérico INTEGER, INTEGER		DECIMAL
"DECIMAL" en la página 326	SYSIBM	Esta función escalar devuelve una representación decimal de una serie de caracteres, con precisión, escala y carácter decimal opcionales.	
	VARCHAR		DECIMAL
	VARCHAR, INTEGER		DECIMAL
	VARCHAR, INTEGER, INTEGER		DECIMAL
	VARCHAR, INTEGER, INTEGER, VARCHAR		DECIMAL
"DECRYPT_BIN y DECRYPT_CHAR" en la página 330	SYSIBM	Esta función escalar devuelve un valor que es el resultado de descifrar datos cifrados utilizando una serie de contraseña.	
	VARCHAR FOR BIT DATA		VARCHAR FOR BIT DATA
	VARCHAR FOR BIT DATA, VARCHAR		VARCHAR FOR BIT DATA
"DECRYPT_BIN y DECRYPT_CHAR" en la página 330	SYSIBM	Esta función escalar devuelve un valor que es el resultado de descifrar datos cifrados utilizando una serie de contraseña.	
	VARCHAR FOR BIT DATA		VARCHAR
	VARCHAR FOR BIT DATA, VARCHAR		VARCHAR
"DEGREES" en la página 332	SYSFUN	Esta función escalar devuelve el número de grados convertidos a partir del argumento expresado en radianes.	
	DOUBLE		DOUBLE
"DEREF" en la página 333	SYSIBM	Esta función escalar devuelve una instancia del tipo de destino del argumento del tipo de referencia.	
	REF(*cualquier-tipo-estructurado*) *con ámbito definido*		*cualquier-tipo-estructurado (igual que tipo de destino de entrada)*

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"DIFFERENCE" en la página 334	SYSFUN	Esta función escalar devuelve la diferencia entre el sonido de las palabras en dos series del argumento, según se haya determinado mediante la función SOUNDEX. El valor 4 significa que las series suenan igual.	
	VARCHAR(4000), VARCHAR(4000)		INTEGER
"DIGITS" en la página 335	SYSIBM	Esta función escalar devuelve la representación en el formato de una serie de caracteres de un número.	
	DECIMAL		CHAR
"DLCOMMENT" en la página 336	SYSIBM	Esta función escalar devuelve el atributo de comentario de un valor DATALINK.	
	DATALINK		VARCHAR(254)
"DLLINKTYPE" en la página 337	SYSIBM	Esta función escalar devuelve el atributo del tipo de enlace de un valor DATALINK.	
	DATALINK		VARCHAR(4)
"DLNEWCOPY" en la página 338	SYSIBM	Esta función escalar devuelve un valor DATALINK que tiene un atributo que indica que se ha modificado el archivo referenciado.	
	VARCHAR(254)		DATALINK
"DLPREVIOUSCOPY" en la página 340	SYSIBM	Esta función escalar devuelve un valor DATALINK que tiene un atributo que indica que debería restaurarse la versión anterior del archivo.	
	VARCHAR(254)		DATALINK
"DLREPLACECONTENT" en la página 342	SYSIBM	Esta función escalar devuelve un valor DATALINK. Cuando la función está en la parte derecha de una cláusula SET en una sentencia UPDATE o está en una cláusula VALUES de una sentencia INSERT, la asignación del valor devuelto da lugar a la sustitución del contenido de un archivo por otro archivo y a la creación de un enlace con el mismo.	
	VARCHAR(200)		DATALINK
	VARCHAR(200), VARCHAR(200)		DATALINK
"DLURLCOMPLETE" en la página 344	SYSIBM	Esta función escalar devuelve el URL completo (incluida la señal de acceso) a partir de un valor DATALINK.	
	DATALINK		VARCHAR
"DLURLCOMPLETEONLY" en la página 345	SYSIBM	Esta función escalar devuelve el URL completo (sin la señal de acceso) a partir de un valor DATALINK con el tipo de enlace de URL.	
	DATALINK		VARCHAR(254)
"DLURLCOMPLETEWRITE" en la página 346	SYSIBM	Esta función escalar devuelve el valor del URL completo necesario para modificar un archivo especificado a partir de un valor DATALINK con el tipo de enlace de URL.	
	DATALINK		VARCHAR(254)
"DLURLPATH" en la página 347	SYSIBM	Esta función escalar devuelve la vía de acceso y el nombre de archivo (incluida señal de acceso) de un valor DATALINK.	
	DATALINK		VARCHAR
"DLURLPATHONLY" en la página 348	SYSIBM	Esta función escalar devuelve la vía de acceso y el nombre de archivo (sin la señal de acceso) de un valor DATALINK.	
	DATALINK		VARCHAR
"DLURLPATHWRITE" en la página 349	SYSIBM	Esta función escalar devuelve la vía de acceso y el nombre de archivo necesarios para modificar un archivo de un servidor determinado a partir de un valor DATALINK con un tipo de enlace de URL.	
	DATALINK		VARCHAR(254)
"DLURLSCHEME" en la página 350	SYSIBM	Esta función escalar devuelve el esquema a partir del atributo del URL de un valor DATALINK.	
	DATALINK		VARCHAR

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"DLURLSERVER" en la página 351	SYSIBM	Esta función escalar devuelve el servidor a partir del atributo del URL de un valor DATALINK.	
	DATALINK		VARCHAR
"DLVALUE" en la página 352	SYSIBM	Esta función escalar crea un valor DATALINK a partir de un argumento ubicación-datos, un argumento tipo de enlace y un argumento serie-comentario opcional.	
	VARCHAR		DATALINK
	VARCHAR, VARCHAR		DATALINK
	VARCHAR, VARCHAR, VARCHAR		DATALINK
"DOUBLE" en la página 354	SYSIBM	Esta función escalar devuelve la representación en el formato de coma flotante de un número.	
	tipo-numérico		DOUBLE
"DOUBLE" en la página 354	SYSFUN	Esta función escalar devuelve el número de coma flotante correspondiente a la representación en el formato de una serie de caracteres de un número. Los blancos iniciales y finales de *argumento* se pasan por alto.	
	VARCHAR		DOUBLE
"ENCRYPT" en la página 356	SYSIBM	Esta función escalar devuelve un valor que es el resultado de cifrar una expresión de serie de datos.	
	VARCHAR		VARCHAR FOR BIT DATA
	VARCHAR, VARCHAR		VARCHAR FOR BIT DATA
	VARCHAR, VARCHAR, VARCHAR		VARCHAR FOR BIT DATA
"EVENT_MON_STATE" en la página 359	SYSIBM	Esta función escalar devuelve el estado operativo de un determinado supervisor de sucesos.	
	VARCHAR		INTEGER
EXEC_DB2_SCRIPT	SYSPROC	Este procedimiento (una rutina de administración de SQL) ejecuta mandatos de DB2 en una instancia de DB2.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"EXP" en la página 360	SYSFUN	Esta función escalar devuelve la función exponencial del argumento.	
	DOUBLE		DOUBLE
"FLOAT" en la página 361	SYSIBM	Esta función escalar es igual que DOUBLE.	
"FLOOR" en la página 362	SYSIBM	Esta función escalar devuelve el entero más grande que es menor o igual que el argumento.	
	SMALLINT		SMALLINT
	INTEGER		INTEGER
	BIGINT		BIGINT
	DOUBLE		DOUBLE
"GENERATE_UNIQUE" en la página 364	SYSIBM	Esta función escalar devuelve una serie de caracteres de datos de bits que es exclusiva comparada con cualquier otra ejecución de la misma función.	
	sin argumento		CHAR(13) FOR BIT DATA
GET_DB_CONFIG	SYSFUN	Esta función de tabla (una rutina de administración de SQL) devuelve la información sobre la configuración de la base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
GET_DBM_CONFIG	SYSFUN	Esta función de tabla (una rutina de administración de SQL) devuelve la información sobre la configuración del gestor de bases de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
GET_DBSIZE_INFO	SYSTOOLS	Este procedimiento (una rutina de administración de SQL) calcula el tamaño de la base de datos y su capacidad máxima.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
GET_ROUTINE_OPTS	SYSPROC	Esta función escalar (una rutina de administración de SQL) devuelve un valor de serie de caracteres de las opciones que deben utilizarse para la creación de procedimientos de SQL en la sesión actual.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
GET_ROUTINE_SAR	SYSFUN	Este procedimiento (una rutina de administración de SQL) devuelve la información necesaria para instalar una rutina idéntica en otro servidor de bases de datos que funcione por lo menos al mismo nivel y con el mismo sistema operativo.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"GETHINT" en la página 363	SYSIBM	Esta función escalar devuelve la contraseña sugerida, si se encuentra una.	
	VARCHAR o CLOB		VARCHAR
"GRAPHIC" en la página 366	SYSIBM	Esta función escalar convierte del tipo fuente a GRAPHIC, con longitud opcional.	
	tipo-gráfico		GRAPHIC
	tipo-gráfico, INTEGER		GRAPHIC
"GROUPING" en la página 281	SYSIBM	Esta función agregada se utiliza con conjuntos-agrupaciones y super-grupos para indicar el subtotal de filas generadas por un conjunto de agrupaciones. El valor devuelto es 0 o 1. Un valor de 1 significa que el valor del argumento de la fila devuelta es un valor nulo y la fila se ha generado para un conjunto de agrupaciones. Esta fila generada proporciona un subtotal correspondiente a un conjunto de agrupaciones.	
	cualquier-tipo		SMALLINT
"HASHEDVALUE" en la página 368 [3]	SYSIBM	Esta función escalar devuelve el índice de correlación de particionamiento (de 0 a 4095) de la fila. El argumento es un nombre de columna dentro de una tabla.	
	cualquier-tipo		INTEGER
HEALTH_DB_HIC	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve la información sobre el indicador de salud de un conjunto a partir de una instantánea de salud de una base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_DB_HIC_HIS	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve la información histórica sobre el indicador de salud de un conjunto a partir de una instantánea de salud de una base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_CONT_HI	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con la información sobre el indicador de salud de los contenedores a partir de una instantánea de salud de una base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_CONT_HI_HIS	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con la información histórica sobre el indicador de salud de los contenedores a partir de una instantánea de salud de una base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
HEALTH_CONT_INFO	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con la información sobre los contenedores a partir de una instantánea de salud de una base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_DB_HI	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con la información sobre el indicador de salud a partir de una instantánea de salud de una base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_DB_HI_HIS	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con la información histórica sobre el indicador de salud a partir de una instantánea de salud de una base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_DB_INFO	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información a partir de una instantánea de salud de una base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_DBM_HI	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con la información sobre el indicador de salud a partir de una instantánea de salud del gestor de bases de datos de DB2.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_DBM_HI_HIS	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con la información histórica sobre el indicador de salud a partir de una instantánea de salud del gestor de bases de datos de DB2.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_DBM_INFO	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información a partir de una instantánea de salud del gestor de bases de datos de DB2.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_HI_REC	SYSPROC	Este procedimiento (una rutina de administración de SQL) recupera un conjunto de recomendaciones que direccionan un indicador de salud en estado de alerta en un objeto de DB2 determinado.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_TBS_HI	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con la información sobre el indicador de salud de los espacios de tablas a partir de una instantánea de salud de una base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_TBS_HI_HIS	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con la información histórica sobre el indicador de salud de los espacios de tablas a partir de una instantánea de salud de una base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
HEALTH_TBS_INFO	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con la información sobre los espacios de tablas a partir de una instantánea de salud de una base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"HEX" en la página 370	SYSIBM	Esta función escalar devuelve la parte hexadecimal de un valor.	
	cualquier-tipo-integrado		VARCHAR
"HOUR" en la página 372	SYSIBM	Esta función escalar devuelve la parte correspondiente a la hora de un valor.	
	VARCHAR		INTEGER
	TIME		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
"IDENTITY_VAL_LOCAL" en la página 373	SYSIBM	Esta función escalar devuelve el valor asignado más reciente correspondiente a una columna de entidad.	
			DECIMAL
"INSERT" en la página 378	SYSFUN	Esta función escalar devuelve una serie en la que *argumento3* bytes se han suprimido de *argumento1* comenzando por *argumento2* y en la que *argumento4* bytes se han insertado en *argumento1* comenzando por *argumento2*.	
	VARCHAR(4000), INTEGER, INTEGER, VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M), INTEGER, INTEGER, CLOB(1M)		CLOB(1M)
	BLOB(1M), INTEGER, INTEGER, BLOB(1M)		BLOB(1M)
INSTALLAPP	DB2EAS	Este procedimiento (una rutina de administración de SQL) instala una aplicación determinada en el servidor de aplicaciones de DB2.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"INTEGER" en la página 379	SYSIBM	Esta función escalar devuelve la representación entera de un número.	
	tipo-numérico		INTEGER
	VARCHAR		INTEGER
"JULIAN_DAY" en la página 381	SYSFUN	Esta función escalar devuelve un valor entero que representa el número de días desde el 1 de enero de 4712 A.C. (la fecha de inicio del calendario Juliano) hasta el valor de fecha especificado en el *argumento*.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
"LCASE (esquema SYSFUN)" en la página 383	SYSFUN	Esta función escalar devuelve una serie en la que todos los caracteres se han convertido a minúsculas. LCASE sólo maneja caracteres del conjunto no variable. Por lo tanto, LCASE(UCASE(serie)) no necesariamente devuelve el mismo resultado que LCASE(serie).	
	VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M)		CLOB(1M)
"LCASE o LOWER" en la página 382	SYSIBM	Esta función escalar devuelve una serie en la que todos los caracteres se han convertido a minúsculas.	
	CHAR		CHAR
	VARCHAR		VARCHAR
"LEFT" en la página 384	SYSFUN	Esta función escalar devuelve una serie que consta de los *argumento2* bytes situados más a la izquierda de *argumento1*.	
	VARCHAR(4000), INTEGER		VARCHAR(4000)
	CLOB(1M), INTEGER		CLOB(1M)
	BLOB(1M), INTEGER		BLOB(1M)
"LENGTH" en la página 385	SYSIBM	Esta función escalar devuelve la longitud del operando en bytes (excepto para los tipos de series de doble byte, que devuelven la longitud en caracteres).	
	cualquier-tipo-integrado		INTEGER

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"LN" en la página 386	SYSFUN	Esta función escalar devuelve el logaritmo natural del argumento (igual que LOG).	
	DOUBLE		DOUBLE
"LOCATE" en la página 387	SYSFUN	Esta función escalar devuelve la posición inicial de la primera ocurrencia de *argumento1* dentro de *argumento2*.Si se especifica el tercer argumento opcional, indica la posición de caracteres en *argumento2* en la que debe comenzar la búsqueda.Si no se encuentra *argumento1* dentro de *argumento2*, se devuelve el valor 0.	
	VARCHAR(4000), VARCHAR(4000)		INTEGER
	VARCHAR(4000), VARCHAR(4000), INTEGER		INTEGER
	CLOB(1M), CLOB(1M)		INTEGER
	CLOB(1M), CLOB(1M), INTEGER		INTEGER
	BLOB(1M), BLOB(1M)		INTEGER
	BLOB(1M), BLOB(1M), INTEGER		INTEGER
"LOG" en la página 388	SYSFUN	Esta función escalar devuelve el logaritmo natural del argumento (igual que LN).	
	DOUBLE		DOUBLE
"LOG10" en la página 389	SYSFUN	Esta función escalar devuelve el logaritmo en base 10 del argumento.	
	DOUBLE		DOUBLE
"LONG_VARCHAR" en la página 390	SYSIBM	Esta función escalar devuelve una serie larga.	
	tipo-carácter		LONG VARCHAR
"LONG_VARGRAPHIC" en la página 391	SYSIBM	Esta función escalar convierte del tipo fuente a LONG VARGRAPHIC.	
	tipo-gráfico		LONG VARGRAPHIC
"LTRIM (esquema SYSFUN)" en la página 393	SYSFUN	Esta función escalar devuelve los caracteres del argumento con los blancos iniciales eliminados.	
	VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M)		CLOB(1M)
"LTRIM" en la página 392	SYSIBM	Esta función escalar devuelve los caracteres del argumento con los blancos iniciales eliminados.	
	CHAR		VARCHAR
	VARCHAR		VARCHAR
	GRAPHIC		VARGRAPHIC
	VARGRAPHIC		VARGRAPHIC
"MAX" en la página 283	SYSIBM	Esta función agregada devuelve el valor máximo de un conjunto de valores.	
	cualquier-tipo-integrado [5]		*igual que tipo de entrada*
"MICROSECOND" en la página 394	SYSIBM	Esta función escalar devuelve la parte correspondiente a las milésimas de segundo (unidad de tiempo) de un valor.	
	VARCHAR		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
"MIDNIGHT_SECONDS" en la página 395	SYSFUN	Esta función escalar devuelve un valor entero comprendido entre 0 y 86 400 que representa el número de segundos transcurridos entre medianoche y el valor de hora especificado en el *argumento*.	
	VARCHAR(26)		INTEGER
	TIME		INTEGER
	TIMESTAMP		INTEGER

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema Parámetros de entrada	Descripción Devuelve
"MIN" en la página 285	SYSIBM	Esta función agregada devuelve el valor mínimo de un conjunto de valores.
	cualquier-tipo-integrado [5]	*igual que tipo de entrada*
"MINUTE" en la página 396	SYSIBM	Esta función escalar devuelve la parte correspondiente a los minutos de un valor.
	VARCHAR	INTEGER
	TIME	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
"MOD" en la página 397	SYSFUN	Esta función escalar devuelve el resto de *argumento1* dividido entre *argumento2*.El resultado sólo es negativo si *argumento1* es negativo.
	SMALLINT, SMALLINT	SMALLINT
	INTEGER, INTEGER	INTEGER
	BIGINT, BIGINT	BIGINT
"MONTH" en la página 398	SYSIBM	Esta función escalar devuelve la parte correspondiente al mes de un valor.
	VARCHAR	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
	DECIMAL	INTEGER
"MONTHNAME" en la página 399	SYSFUN	Esta función escalar devuelve una serie que combina caracteres en mayúsculas y minúsculas que contiene el nombre del mes (por ejemplo, enero) de la parte correspondiente al mes del argumento, según el entorno local en el momento en que se inició la base de datos.
	VARCHAR(26)	VARCHAR(100)
	DATE	VARCHAR(100)
	TIMESTAMP	VARCHAR(100)
MQPUBLISH	DB2MQ, DB2MQ1C	Esta función escalar (una rutina de administración de SQL) publica los datos de una ubicación MQSeries.
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.	
MQREAD	DB2MQ, DB2MQ1C	Esta función escalar (una rutina de administración de SQL) devuelve un mensaje a partir de una ubicación MQSeries.
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.	
MQREADALL	DB2MQ, DB2MQ1C	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con mensajes y metadatos de los mensajes a partir de una ubicación MQSeries.
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.	
MQREADALLCLOB	DB2MQ	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla que contiene mensajes y metadatos de los mensajes a partir de una ubicación MQSeries determinada.
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.	
MQREADCLOB	DB2MQ	Esta función escalar (una rutina de administración de SQL) devuelve un mensaje a partir de una ubicación MQSeries determinada.
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.	

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
MQRECEIVE	DB2MQ, DB2MQ1C	Esta función escalar (una rutina de administración de SQL) devuelve un mensaje a partir de una ubicación MQSeries y elimina el mensaje de la cola asociada.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
MQRECEIVEALL	DB2MQ, DB2MQ1C	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla que contiene mensajes y metadatos de los mensajes a partir de una ubicación MQSeries y elimina los mensajes de la cola asociada.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
MQRECEIVEALLCLOB	DB2MQ	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla que contiene mensajes y metadatos de los mensajes a partir de una ubicación MQSeries determinada.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
MQRECEIVECLOB	DB2MQ	Esta función escalar (una rutina de administración de SQL) devuelve un mensaje a partir de una ubicación MQSeries determinada.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
MQSEND	DB2MQ, DB2MQ1C	Esta función escalar (una rutina de administración de SQL) envía los datos a una ubicación MQSeries.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
MQSUBSCRIBE	DB2MQ, DB2MQ1C	Esta función escalar (una rutina de administración de SQL) permite suscribirse a los mensajes de MSQseries publicados sobre un tema determinado.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
MQUNSUBSCRIBE	DB2MQ, DB2MQ1C	Esta función escalar (una rutina de administración de SQL) cancela la suscripción a los mensajes de MSQseries publicados sobre un tema determinado.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"MULTIPLY_ALT" en la página 400	SYSIBM	Esta función escalar devuelve el producto de dos argumentos como un valor decimal. Esta función resulta útil cuando la suma de las precisiones del argumento es mayor que 31.	
	tipo-numérico-exacto, *tipo-numérico-exacto*		DECIMAL
"NULLIF" en la página 402 [3]	SYSIBM	Esta función escalar devuelve NULL si los argumentos son iguales o devuelve el primer argumento si no lo son.	
	cualquier-tipo [5], *cualquier-tipo-comparable*[5]		*cualquier-tipo*
"POSSTR" en la página 403	SYSIBM	Esta función escalar devuelve la posición en la que una serie está contenida en otra.	
	tipo-serie, *tipo-serie-compatible*		INTEGER
"POWER" en la página 405	SYSFUN	Esta función escalar devuelve el valor de *argumento1* elevado a la potencia de *argumento2*.	
	INTEGER, INTEGER		INTEGER
	BIGINT, BIGINT		BIGINT
	DOUBLE, INTEGER		DOUBLE
	DOUBLE, DOUBLE		DOUBLE
PUT_ROUTINE_SAR	SYSFUN	Este procedimiento (una rutina de administración de SQL) pasa la información necesaria para crear y definir una rutina SQL en el servidor de bases de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema Parámetros de entrada	Descripción Devuelve
"QUARTER" en la página 406	SYSFUN	Esta función escalar devuelve un valor entero comprendido entre 1 y 4 que representa el trimestre del año correspondiente a la fecha especificada en el argumento.
	VARCHAR(26)	INTEGER
	DATE	INTEGER
	TIMESTAMP	INTEGER
"RADIANS" en la página 407	SYSFUN	Esta función escalar devuelve el número de radianes convertidos a partir del argumento expresado en grados.
	DOUBLE	DOUBLE
"RAISE_ERROR" en la página 408 [3]	SYSIBM	Esta función escalar envía un error a la SQLCA. El sqlstate que se devolverá se indica mediante *argumento1*. El segundo argumento contiene el texto que debe devolverse.
	VARCHAR, VARCHAR	*cualquier-tipo* [6]
"RAND" en la página 409	SYSFUN	Esta función escalar devuelve un valor aleatorio de coma flotante comprendido entre 0 y 1 utilizando el argumento como valor generador opcional.
	no se necesita ningún argumento	DOUBLE
	INTEGER	DOUBLE
"REAL" en la página 410	SYSIBM	Esta función escalar devuelve la representación en el formato de coma flotante de precisión simple de un número.
	tipo-numérico	REAL
REBIND_ROUTINE_PACKAGE	SYSPROC	Este procedimiento (una rutina de administración de SQL) vuelve a vincular el paquete asociado con un procedimiento de SQL.
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.	
"REC2XML" en la página 411	SYSIBM	Esta función escalar devuelve una serie formateada codificada en XML que contiene nombres de columna y datos de columna.
	DECIMAL, VARCHAR, VARCHAR, *cualquier-tipo*[7]	VARCHAR
"Funciones de regresión" en la página 286	SYSIBM	La función agregada REGR_AVGX devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
	tipo-numérico, *tipo-numérico*	DOUBLE
"Funciones de regresión" en la página 286	SYSIBM	La función agregada REGR_AVGY devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
	tipo-numérico, *tipo-numérico*	DOUBLE
"Funciones de regresión" en la página 286	SYSIBM	La función agregada REGR_COUNT devuelve el número de pares de números no nulos utilizados para acomodar la línea de regresión.
	tipo-numérico, *tipo-numérico*	INTEGER
"Funciones de regresión" en la página 286	SYSIBM	La función agregada REGR_INTERCEPT o REGR_ICPT devuelve la intersección y de la línea de regresión.
	tipo-numérico, *tipo-numérico*	DOUBLE
"Funciones de regresión" en la página 286	SYSIBM	La función agregada REGR_R2 devuelve el coeficiente de determinación de la regresión.
	tipo-numérico, *tipo-numérico*	DOUBLE
"Funciones de regresión" en la página 286	SYSIBM	La función agregada REGR_SLOPE devuelve la inclinación de la línea.
	tipo-numérico, *tipo-numérico*	DOUBLE
"Funciones de regresión" en la página 286	SYSIBM	La función agregada REGR_SXX devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
	tipo-numérico, *tipo-numérico*	DOUBLE

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"Funciones de regresión" en la página 286	SYSIBM	La función agregada REGR_SXY devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.	
	tipo-numérico, *tipo-numérico*		DOUBLE
"Funciones de regresión" en la página 286	SYSIBM	La función agregada REGR_SYY devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.	
	tipo-numérico, *tipo-numérico*		DOUBLE
REORGCHK_IX_STATS	SYSPROC	Este procedimiento (una rutina de administración de SQL) comprueba las estadísticas del índice para determinar si es o no es necesaria una reorganización.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
REORGCHK_TB_STATS	SYSPROC	Este procedimiento (una rutina de administración de SQL) comprueba las estadísticas de la tabla para determinar si es o no es necesaria una reorganización.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"REPEAT" en la página 416	SYSFUN	Esta función escalar devuelve una serie de caracteres formada por el *argumento1* repetido *argumento2* veces.	
	VARCHAR(4000), INTEGER		VARCHAR(4000)
	CLOB(1M), INTEGER		CLOB(1M)
	BLOB(1M), INTEGER		BLOB(1M)
"REPLACE" en la página 417	SYSFUN	Esta función escalar sustituye todas las ocurrencias de *argumento2* en *argumento1* por *argumento3*.	
	VARCHAR(4000), VARCHAR(4000), VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M), CLOB(1M), CLOB(1M)		CLOB(1M)
	BLOB(1M), BLOB(1M), BLOB(1M)		BLOB(1M)
"RIGHT" en la página 418	SYSFUN	Esta función escalar devuelve una serie que consta de los *argumento2* bytes situados más a la derecha de *argumento1*.	
	VARCHAR(4000), INTEGER		VARCHAR(4000)
	CLOB(1M), INTEGER		CLOB(1M)
	BLOB(1M), INTEGER		BLOB(1M)
"ROUND" en la página 419	SYSIBM	Esta función escalar devuelve el primer argumento redondeado a *argumento2* posiciones a la derecha de la coma decimal. Si *argumento2* es negativo, *argumento1* se redondea al valor absoluto de *argumento2* posiciones a la izquierda de la coma decimal.	
	INTEGER, INTEGER		INTEGER
	BIGINT, INTEGER		BIGINT
	DOUBLE, INTEGER		DOUBLE
"RTRIM (esquema SYSFUN)" en la página 422	SYSFUN	Esta función escalar devuelve los caracteres del argumento con los blancos de cola eliminados.	
	VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M)		CLOB(1M)
"RTRIM" en la página 421	SYSIBM	Esta función escalar devuelve los caracteres del argumento con los blancos de cola eliminados.	
	CHAR		VARCHAR
	VARCHAR		VARCHAR
	GRAPHIC		VARGRAPHIC
	VARGRAPHIC		VARGRAPHIC

Funciones soportadas

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"SECOND" en la página 423	SYSIBM	Esta función escalar devuelve la parte correspondiente a los segundos (unidad de tiempo) de un valor.	
	VARCHAR		INTEGER
	TIME		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
SERVER	DB2EAS	Este procedimiento (una rutina de administración de SQL) inicia o detiene el servidor de aplicaciones de DB2.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SET_ROUTINE_OPTS	SYSPROC	Este procedimiento (una rutina de administración de SQL) establece las opciones que deben utilizarse para la creación de procedimientos de SQL en la sesión actual.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"SIGN" en la página 424	SYSFUN	Esta función escalar devuelve un indicador del signo del argumento. Si el argumento es menor que cero, se devuelve -1. Si el argumento es igual a cero, se devuelve 0. Si el argumento es mayor que cero, se devuelve 1.	
	SMALLINT		SMALLINT
	INTEGER		INTEGER
	BIGINT		BIGINT
	DOUBLE		DOUBLE
"SIN" en la página 425	SYSIBM	Esta función escalar devuelve el seno del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
"SINH SINH" en la página 426	SYSIBM	Esta función escalar devuelve el seno hiperbólico del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
"SMALLINT" en la página 427	SYSIBM	Esta función escalar devuelve la representación en el formato de un entero pequeño de un número.	
	tipo-numérico		SMALLINT
	VARCHAR		SMALLINT
SNAPSHOT_AGENT	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información sobre los agentes a partir de una instantánea de la aplicación.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_APPL	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información general a partir de una instantánea de la aplicación.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_APPL_INFO	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información general a partir de una instantánea de la aplicación.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_BP	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información a partir de una instantánea de la agrupación de almacenamientos intermedios.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
SNAPSHOT_CONTAINER	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información sobre la configuración del contenedor a partir de una instantánea del espacio de tablas.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_DATABASE	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información a partir de una instantánea de la base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_DBM	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información a partir de una instantánea del gestor de bases de datos de DB2.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_DYN_SQL	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información a partir de una instantánea del SQL dinámico.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_FCM	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información a nivel del gestor de bases de datos respecto a Fast Communication Manager.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_FCMNODE	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información a partir de una instantánea del Fast Communication Manager del gestor de bases de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_FILEW	SYSPROC	Este procedimiento (una rutina de administración de SQL) graba los datos de una instantánea del sistema en un archivo situado en el subdirectorio `tmp` del directorio de la instancia.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_LOCK	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información a partir de una instantánea del bloqueo.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_LOCKWAIT	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información sobre las esperas de bloqueo a partir de una instantánea de la aplicación.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_QUIESCERS	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información sobre los paralizadores a partir de una instantánea del espacio de tablas.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_RANGES	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información a partir de una instantánea del rango.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
SNAPSHOT_STATEMENT	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información sobre las sentencias a partir de una instantánea de la aplicación.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_SUBSECT	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información sobre las subsecciones de los planes de acceso a partir de una instantánea de la aplicación.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_SWITCHES	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información sobre el estado de conmutación de una instantánea de la base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_TABLE	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información sobre la actividad a partir de una instantánea de la tabla.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_TBREORG	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información sobre la reorganización de la tabla.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_TBS	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información sobre la actividad a partir de una instantánea del espacio de tablas.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SNAPSHOT_TBS_CFG	SYSPROC	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con información sobre la configuración a partir de una instantánea del espacio de tablas.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"SOUNDEX" en la página 428	SYSFUN	Esta función escalar devuelve un código de 4 caracteres que representa el sonido de las palabras del argumento. Este resultado se puede comparar con el sonido de otras series.	
	VARCHAR(4000)		CHAR(4)
"SPACE" en la página 429	SYSFUN	Esta función escalar devuelve una serie de caracteres formada por *argumento1* blancos.	
	INTEGER		VARCHAR(4000)
SQLCACHE_SNAPSHOT	SYSFUN	Esta función de tabla (una rutina de administración de SQL) devuelve una tabla con una instantánea de la antememoria de la sentencia de SQL dinámico de DB2.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"SQRT" en la página 430	SYSFUN	Esta función escalar devuelve la raíz cuadrada del argumento.	
	DOUBLE		DOUBLE
"STDDEV" en la página 289	SYSIBM	Esta función agregada devuelve la desviación estándar de un conjunto de números.	
	DOUBLE		DOUBLE

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"SUBSTR" en la página 431	SYSIBM	Esta función escalar devuelve una subserie de la serie *argumento1*, empezando por *argumento2*. La subserie tiene una longitud de *argumento3* caracteres. Si no se especifica *argumento3*, se da por supuesto el resto de la serie.	
	tipo-serie, INTEGER		*tipo-serie*
	tipo-serie, INTEGER, INTEGER		*tipo-serie*
"SUM" en la página 290	SYSIBM	Esta función agregada devuelve la suma de un conjunto de números.	
	tipo-numérico[4]		*tipo-numérico-máx*[1]
SYSINSTALLOBJECTS	SYSPROC	Este procedimiento (una rutina de administración de SQL) crea o descarta los objetos de la base de datos que necesita una herramienta determinada.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
SYSINSTALLROUTINES	SYSPROC	Este procedimiento (una rutina de administración de SQL) crea nuevos procedimientos y funciones en el esquema SYSPROC si el programa de utilidad **db2updv8** no se ha ejecutado con posterioridad a la migración de la base de datos.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"TABLE_NAME" en la página 434	SYSIBM	Esta función escalar devuelve un nombre no calificado de una tabla o vista basado en el nombre de objeto especificado en *argumento1* y en el nombre de esquema opcional especificado en *argumento2*. El valor devuelto se utiliza para resolver los alias.	
	VARCHAR		VARCHAR(128)
	VARCHAR, VARCHAR		VARCHAR(128)
"TABLE_SCHEMA" en la página 435	SYSIBM	Esta función escalar devuelve la parte correspondiente al nombre de esquema de un nombre de tabla o de vista de dos partes (especificado por el nombre del objeto en *argumento1* y por el nombre de esquema opcional en *argumento2*). El valor devuelto se utiliza para resolver los alias.	
	VARCHAR		VARCHAR(128)
	VARCHAR, VARCHAR		VARCHAR(128)
"TAN" en la página 437	SYSIBM	Esta función escalar devuelve la tangente del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
"TANH" en la página 438	SYSIBM	Esta función escalar devuelve la tangente hiperbólica del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
"TIME" en la página 439	SYSIBM	Esta función escalar devuelve una hora a partir de un valor.	
	TIME		TIME
	TIMESTAMP		TIME
	VARCHAR		TIME
"TIMESTAMP" en la página 440	SYSIBM	Esta función escalar devuelve una indicación de fecha y hora a partir de un valor o de un par de valores.	
	TIMESTAMP		TIMESTAMP
	VARCHAR		TIMESTAMP
	VARCHAR, VARCHAR		TIMESTAMP
	VARCHAR, TIME		TIMESTAMP
	DATE, VARCHAR		TIMESTAMP
	DATE, TIME		TIMESTAMP

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"TIMESTAMP_FORMAT" en la página 441	SYSIBM	Esta función escalar devuelve una indicación de fecha y hora a partir de una serie de caracteres (*argumento1*) que se ha interpretado utilizando una plantilla de formato (*argumento2*).	
	VARCHAR, VARCHAR		TIMESTAMP
"TIMESTAMP_ISO" en la página 443	SYSFUN	Esta función escalar devuelve un valor de indicación de fecha y hora basado en un argumento de fecha, de hora o de indicación de fecha y hora. Si el argumento es una fecha, inserta cero para todos los elementos de hora. Si el argumento es una hora, inserta el valor de CURRENT DATE para los elementos de fecha y ceros para el elemento de fracción de hora.	
	DATE		TIMESTAMP
	TIME		TIMESTAMP
	TIMESTAMP		TIMESTAMP
	VARCHAR(26)		TIMESTAMP
"TIMESTAMPDIFF" en la página 444	SYSFUN	Esta función escalar devuelve un número estimado de intervalos de tipo *argumento1* basado en la diferencia entre dos indicaciones de fecha y hora. El segundo argumento es el resultado de restar dos tipos de indicaciones de fecha y hora y de convertir el resultado en CHAR. Los tipos de intervalos válidos son los siguientes: **1** Fracciones de un segundo **2** Segundos **4** Minutos **8** Horas **16** Días **32** Semanas **64** Meses **128** Trimestres **256** Años	
	INTEGER, CHAR(22)		INTEGER
"TO_CHAR" en la página 446	SYSIBM	Esta función escalar devuelve una representación en el formato de caracteres de una indicación de fecha y hora.	
	Igual que VARCHAR_FORMAT.		Igual que VARCHAR_FORMAT.
"TO_DATE" en la página 447	SYSIBM	Esta función escalar devuelve una indicación de la fecha y hora a partir de una serie de caracteres.	
	Igual que TIMESTAMP_FORMAT.		Igual que TIMESTAMP_FORMAT.

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"TRANSLATE" en la página 448	SYSIBM	Esta función escalar devuelve una serie en la que uno o más caracteres pueden haberse convertido a otros caracteres.	
	CHAR		CHAR
	VARCHAR		VARCHAR
	CHAR, VARCHAR, VARCHAR		CHAR
	VARCHAR, VARCHAR, VARCHAR		VARCHAR
	CHAR, VARCHAR, VARCHAR, VARCHAR		CHAR
	VARCHAR, VARCHAR, VARCHAR, VARCHAR		VARCHAR
	GRAPHIC, VARGRAPHIC, VARGRAPHIC		GRAPHIC
	VARGRAPHIC, VARGRAPHIC, VARGRAPHIC		VARGRAPHIC
	GRAPHIC, VARGRAPHIC, VARGRAPHIC, VARGRAPHIC		GRAPHIC
	VARGRAPHIC, VARGRAPHIC, VARGRAPHIC, VARGRAPHIC		VARGRAPHIC
"TRUNCATE o TRUNC" en la página 450	SYSIBM	Esta función escalar devuelve el *argumento1* truncado a *argumento2* posiciones a la derecha de la coma decimal. Si *argumento2* es negativo, *argumento1* se trunca al valor absoluto de *argumento2* posiciones a la izquierda de la coma decimal.	
	INTEGER, INTEGER		INTEGER
	BIGINT, INTEGER		BIGINT
	DOUBLE, INTEGER		DOUBLE
"TYPE_ID" en la página 451 [3]	SYSIBM	Esta función escalar devuelve el identificador de tipo de datos interno del tipo de datos dinámico del argumento. Tenga en cuenta que el resultado de esta función no se puede portar entre bases de datos.	
	cualquier-tipo-estructurado		INTEGER
"TYPE_NAME" en la página 452 [3]	SYSIBM	Esta función escalar devuelve el nombre no calificado del tipo de datos dinámico del argumento.	
	cualquier-tipo-estructurado		VARCHAR(18)
"TYPE_SCHEMA" en la página 453 [3]	SYSIBM	Esta función escalar devuelve el nombre de esquema del tipo dinámico del argumento.	
	cualquier-tipo-estructurado		VARCHAR(128)
"UCASE o UPPER" en la página 454	SYSFUN	Esta función escalar devuelve una serie en la que todos los caracteres se han convertido a mayúsculas.	
	VARCHAR		VARCHAR
"UCASE o UPPER" en la página 454	SYSIBM	Esta función escalar devuelve una serie en la que todos los caracteres se han convertido a mayúsculas.	
	CHAR		CHAR
	VARCHAR		VARCHAR
UNINSTALLAPP	DB2EAS	Este procedimiento (una rutina de administración de SQL) desinstala una aplicación determinada del servidor de aplicaciones de DB2.	
	En el Centro de información de DB2 hallará una descripción completa de esta rutina de administración de SQL.		
"VALUE" en la página 455 [3]	SYSIBM	Esta función escalar es igual que COALESCE.	
"VARCHAR" en la página 456	SYSIBM	Esta función escalar devuelve una representación en el formato VARCHAR del primer argumento. Si existe un segundo argumento, especifica la longitud del resultado.	
	tipo-carácter		VARCHAR
	tipo-carácter, INTEGER		VARCHAR
	tipo-fechahora		VARCHAR

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema	Descripción	
	Parámetros de entrada		**Devuelve**
"VARCHAR_FORMAT" en la página 458	SYSIBM	Esta función escalar devuelve una representación en el formato de caracteres de una indicación de fecha y hora (*argumento1*) con el formato indicado en una plantilla (*argumento2*).	
	TIMESTAMP, VARCHAR		VARCHAR
	VARCHAR, VARCHAR		VARCHAR
"VARGRAPHIC" en la página 460	SYSIBM	Esta función escalar devuelve una representación en el formato VARGRAPHIC del primer argumento. Si existe un segundo argumento, especifica la longitud del resultado.	
	tipo-gráfico		VARGRAPHIC
	tipo-gráfico, INTEGER		VARGRAPHIC
	VARCHAR		VARGRAPHIC
"VARIANCE" en la página 291	SYSIBM	Esta función agregada devuelve la varianza de un conjunto de números.	
	DOUBLE		DOUBLE
"WEEK" en la página 462	SYSFUN	Esta función escalar devuelve la semana del año del argumento como un valor entero comprendido entre 1 y 54.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
"WEEK_ISO" en la página 463	SYSFUN	Esta función escalar devuelve la semana del año del argumento como un valor entero comprendido entre 1 y 53. El primer día de una semana es el lunes. La semana 1 es la primera semana del año que contiene un jueves.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
"YEAR" en la página 464	SYSIBM	Esta función escalar devuelve la parte correspondiente al año de un valor.	
	VARCHAR		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
"+"	SYSIBM	Suma dos operandos numéricos.	
	tipo-numérico, *tipo-numérico*		*tipo-numérico-máx*
"+"	SYSIBM	Operatorio más unario.	
	tipo-numérico		*tipo-numérico*
"+"	SYSIBM	Operador más de fecha y hora.	
	DATE, DECIMAL(8,0)		DATE
	TIME, DECIMAL(6,0)		TIME
	TIMESTAMP, DECIMAL(20,6)		TIMESTAMP
	DECIMAL(8,0), DATE		DATE
	DECIMAL(6,0), TIME		TIME
	DECIMAL(20,6), TIMESTAMP		TIMESTAMP
	tipo-fechahora, DOUBLE, *código-duración-etiquetado*		*tipo-fechahora*
"–"	SYSIBM	Resta dos operandos numéricos.	
	tipo-numérico, *tipo-numérico*		*tipo-numérico-máx*
"–"	SYSIBM	Operatorio menos unario.	
	tipo-numérico		*tipo-numérico* [1]

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema Parámetros de entrada	Descripción Devuelve
"–"	SYSIBM	Operador menos de fecha y hora.
	DATE, DATE	DECIMAL(8,0)
	TIME, TIME	DECIMAL(6,0)
	TIMESTAMP, TIMESTAMP	DECIMAL(20,6)
	DATE, VARCHAR	DECIMAL(8,0)
	TIME, VARCHAR	DECIMAL(6,0)
	TIMESTAMP, VARCHAR	DECIMAL(20,6)
	VARCHAR, DATE	DECIMAL(8,0)
	VARCHAR, TIME	DECIMAL(6,0)
	VARCHAR, TIMESTAMP	DECIMAL(20,6)
	DATE, DECIMAL(8,0)	DATE
	TIME, DECIMAL(6,0)	TIME
	TIMESTAMP, DECIMAL(20,6)	TIMESTAMP
	tipo-fechahora, DOUBLE, *código-duración-etiquetado*	*tipo-fechahora*
"*"	SYSIBM	Multiplica dos operandos numéricos.
	tipo-numérico, *tipo-numérico*	*tipo-numérico-máx*
"/"	SYSIBM	Divide dos operandos numéricos.
	tipo-numérico, *tipo-numérico*	*tipo-numérico-máx*
"||"	SYSIBM	Igual que CONCAT.

Notas

- En las referencias a tipos de datos de serie que no están calificados por una longitud se da por supuesto que dan soporte a la longitud máxima correspondiente al tipo de datos.
- En las referencias a un tipo de datos DECIMAL sin precisión y escala se da por supuesto que permiten cualquier precisión y escala soportadas.

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema Parámetros de entrada	Descripción Devuelve

Claves de las tablas

cualquier-tipo-integrado
Cualquier tipo de datos que no es un tipo diferenciado.

cualquier-tipo Cualquier tipo definido ante la base de datos.

cualquier-tipo-estructurado
Cualquier tipo estructurado definido por el usuario ante la base de datos.

cualquier-tipo-comparable
Cualquier tipo que se puede comparar con otros tipos de argumento según lo definido en "Asignaciones y comparaciones" en la página 110 .

cualquier-tipo-compatible-unión
Cualquier tipo que sea compatible con otros tipos de argumentos, según lo definido en "Reglas para los tipos de datos del resultado" en la página 125 .

tipo-carácter Cualquiera de los siguientes tipos de series de caracteres: CHAR, VARCHAR, LONG VARCHAR, CLOB.

tipo-serie-compatible
Un tipo de serie que proviene de la misma agrupación que el otro argumento (por ejemplo, si un argumento es un *tipo-carácter* el otro también debe ser un *tipo-carácter*).

tipo-fechahora Cualquiera de los siguientes tipos de fecha y hora: DATE, TIME, TIMESTAMP.

tipo-numérico-exacto
Cualquiera de los tipos numéricos exactos: SMALLINT, INTEGER, BIGINT, DECIMAL.

tipo-gráfico Cualquiera de los siguientes tipos de series de caracteres de doble byte: GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB.

código-duración-etiquetado
Como tipo es SMALLINT. Si la función se invoca utilizando el formato infix del operador más o menos, se pueden utilizar las duraciones etiquetadas según lo definido en "Expresiones". Para una función fuente que no utiliza el carácter del operador más o menos como el nombre, se deben utilizar los siguientes valores para el argumento código-duración-etiquetado cuando se invoca la función.

1	YEAR o YEARS
2	MONTH o MONTHS
3	DAY o DAYS
4	HOUR o HOURS
5	MINUTE o MINUTES
6	SECOND o SECONDS
7	MICROSECOND o MICROSECONDS

tipo-LOB Cualquiera de los tipos de objetos grandes: BLOB, CLOB, DBCLOB.

tipo-numérico-máx
El tipo numérico máximo de los argumentos, donde máximo es el *tipo-numérico* que hay más a la derecha.

tipo-serie-máx El tipo de serie máximo de los argumentos, donde máximo es el *tipo-carácter* o *tipo-gráfico* que hay más a la derecha.Si los argumentos son BLOB, el *tipo-serie-máx* es BLOB.

tipo-numérico Cualquiera de los siguientes tipos numéricos: SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE.

tipo-serie Cualquier tipo desde *tipo-caracteres*, *tipo-gráfico* o BLOB.

Tabla 16. Funciones soportadas (continuación)

Nombre de la función	Esquema Parámetros de entrada	Descripción Devuelve
Notas al pie de las tablas		
[1]	Cuando el parámetro de entrada es SMALLINT, el tipo de resultado es INTEGER. Cuando el parámetro de entrada es REAL, el tipo de resultado es DOUBLE.	
[2]	Las palabras clave permitidas son ISO, USA, EUR, JIS y LOCAL. Esta signatura de función no recibe soporte como una función fuente.	
[3]	Esta función no se puede utilizar como una función fuente.	
[4]	La palabra clave ALL o DISTINCT se puede utilizar antes del primer argumento. Si se especifica DISTINCT, el uso de tipos estructurados definidos por el usuario, tipos de series largos o el tipo DATALINK no recibe soporte.	
[5]	El uso de tipos estructurados definidos por el usuario, tipos de series largos o de un tipo DATALINK no recibe soporte.	
[6]	El tipo devuelto por RAISE_ERROR depende el contexto de su uso. RAISE_ERROR, si no se convierte en un determinado tipo, devolverá un tipo adecuado a su invocación dentro de una expresión CASE.	
[7]	El uso del *tipo-gráfico*, del *tipo-LOB*, de los tipos de series largos y de los tipos DATALINK no recibe soporte.	

Tabla 17. Funciones agregadas

Función	Descripción
"AVG" en la página 274	Devuelve el promedio de un conjunto de números.
"CORRELATION" en la página 276	Devuelve el coeficiente de correlación de un conjunto de pares de números.
"COUNT" en la página 277	Devuelve el número de filas o de valores de un conjunto de filas o de valores.
"COUNT_BIG" en la página 278	Devuelve el número de filas o de valores de un conjunto de filas o de valores. El resultado puede ser mayor que el valor máximo de INTEGER.
"COVARIANCE" en la página 280	Devuelve la covarianza de un conjunto de pares de números.
"GROUPING" en la página 281	Se utiliza con conjuntos-agrupaciones y super-grupos para indicar el subtotal de filas generadas por un conjunto de agrupaciones. El valor devuelto es 0 o 1. Un valor de 1 significa que el valor del argumento de la fila devuelta es un valor nulo y la fila se ha generado para un conjunto de agrupaciones. Esta fila generada proporciona un subtotal correspondiente a un conjunto de agrupaciones.
"MAX" en la página 283	Devuelve el valor máximo de un conjunto de valores.
"MIN" en la página 285	Devuelve el valor mínimo de un conjunto de valores.
"Funciones de regresión" en la página 286	La función agregada REGR_AVGX devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
"Funciones de regresión" en la página 286	La función agregada REGR_AVGY devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
"Funciones de regresión" en la página 286	La función agregada REGR_COUNT devuelve el número de pares de números no nulos utilizados para acomodar la línea de regresión.
"Funciones de regresión" en la página 286	La función agregada REGR_INTERCEPT o REGR_ICPT devuelve la intersección y de la línea de regresión.
"Funciones de regresión" en la página 286	La función agregada REGR_R2 devuelve el coeficiente de determinación de la regresión.

Tabla 17. Funciones agregadas (continuación)

Función	Descripción
"Funciones de regresión" en la página 286	La función agregada REGR_SLOPE devuelve la inclinación de la línea.
"Funciones de regresión" en la página 286	La función agregada REGR_SXX devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
"Funciones de regresión" en la página 286	La función agregada REGR_SXY devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
"Funciones de regresión" en la página 286	La función agregada REGR_SYY devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
"STDDEV" en la página 289	Devuelve la desviación estándar de un conjunto de números.
"SUM" en la página 290	Devuelve la suma de un conjunto de números.
"VARIANCE" en la página 291	Devuelve la varianza de un conjunto de números.

Tabla 18. Funciones escalares de difusión

Función	Descripción
"BIGINT" en la página 300	Devuelve una representación en el formato de un entero de 64 bits de un número o una serie de caracteres en el formato de una constante entera.
"BLOB" en la página 302	Devuelve una representación BLOB de una serie de cualquier tipo.
"CHAR" en la página 304	Devuelve una representación CHARACTER de un valor.
"CLOB" en la página 310	Devuelve una representación CLOB de un valor.
"DATE" en la página 316	Devuelve una representación DATE de un valor.
"DBCLOB" en la página 323	Devuelve una representación DBCLOB de una serie.
"DECIMAL" en la página 326	Devuelve una representación DECIMAL de un número.
"DOUBLE" en la página 354	Devuelve una representación DOUBLE PRECISION de un número.
"FLOAT" en la página 361	Devuelve una representación FLOAT de un número.
"GRAPHIC" en la página 366	Devuelve una representación GRAPHIC de una serie.
"INTEGER" en la página 379	Devuelve una representación INTEGER de un número.
"LONG_VARCHAR" en la página 390	Devuelve una representación LONG VARCHAR de un valor.
"LONG_VARGRAPHIC" en la página 391	Devuelve una representación LONG VARGRAPHIC de un valor.
"REAL" en la página 410	Devuelve una representación REAL de un número.
"SMALLINT" en la página 427	Devuelve una representación SMALLINT de un número.
"TIME" en la página 439	Devuelve una representación TIME de un valor.
"TIMESTAMP" en la página 440	Devuelve una representación TIMESTAMP a partir de un valor o de un par de valores.
"VARCHAR" en la página 456	Devuelve una representación VARCHAR de un valor.
"VARGRAPHIC" en la página 460	Devuelve una representación VARGRAPHIC de un valor.

Tabla 19. Funciones escalares de enlace de datos

Función	Descripción
"DLCOMMENT" en la página 336	Devuelve el atributo de comentario de un valor DATALINK.
"DLLINKTYPE" en la página 337	Devuelve el atributo del tipo de enlace de un valor DATALINK.
"DLNEWCOPY" en la página 338	Devuelve un valor DATALINK que tiene un atributo que indica que se ha modificado el archivo referenciado.
"DLPREVIOUSCOPY" en la página 340	Devuelve un valor DATALINK que tiene un atributo que indica que debería restaurarse la versión anterior del archivo.
"DLREPLACECONTENT" en la página 342	Devuelve un valor DATALINK. Cuando la función está en la parte derecha de una cláusula SET en una sentencia UPDATE o está en una cláusula VALUES de una sentencia INSERT, la asignación del valor devuelto da lugar a la sustitución del contenido de un archivo por otro archivo y a la creación de un enlace con el mismo.
"DLURLCOMPLETE" en la página 344	Devuelve el URL completo (incluida la señal de acceso) de un valor DATALINK.
"DLURLCOMPLETEONLY" en la página 345	Devuelve el URL completo (sin la señal de acceso) de un valor DATALINK con el tipo de enlace de URL.
"DLURLCOMPLETEWRITE" en la página 346	Devuelve el valor del URL completo necesario para modificar un archivo especificado a partir de un valor DATALINK con el tipo de enlace de URL.
"DLURLPATH" en la página 347	Devuelve la vía de acceso y el nombre de archivo (incluida señal de acceso) de un valor DATALINK.
"DLURLPATHONLY" en la página 348	Devuelve la vía de acceso y el nombre de archivo (sin ninguna señal de acceso) de un valor DATALINK.
"DLURLPATHWRITE" en la página 349	Devuelve la vía de acceso y el nombre de archivo necesarios para modificar un archivo dentro de un determinado servidor, a partir de un valor DATALINK con el tipo de enlace de URL.
"DLURLSCHEME" en la página 350	Devuelve el esquema a partir del atributo del URL de un valor DATALINK.
"DLURLSERVER" en la página 351	Devuelve el servidor a partir del atributo del URL de un valor DATALINK.
"DLVALUE" en la página 352	Crea un valor DATALINK a partir de un argumento de ubicación-datos, un argumento de tipo de enlace y un argumento opcional de serie-comentario.

Tabla 20. Funciones escalares de fecha y hora

Función	Descripción
"DAY" en la página 317	Devuelve la parte del día del valor.
"DAYNAME" en la página 318	Devuelve una serie que combina caracteres en mayúsculas y minúsculas que contiene el nombre del día (por ejemplo, viernes) de la parte correspondiente al día del argumento, según el entorno local en el momento en que se emitió db2start.
"DAYOFWEEK" en la página 319	Devuelve el día de la semana a partir de un valor, donde el 1 es el domingo y el 7 es el sábado.

Tabla 20. Funciones escalares de fecha y hora (continuación)

Función	Descripción
"DAYOFWEEK_ISO" en la página 320	Devuelve el día de la semana a partir de un valor, donde el 1 es el lunes y el 7 es el domingo.
"DAYOFYEAR" en la página 321	Devuelve el día del año a partir de un valor.
"DAYS" en la página 322	Devuelve una representación entera de una fecha.
"HOUR" en la página 372	Devuelve la parte de la hora de un valor.
"JULIAN_DAY" en la página 381	Devuelve un valor entero que representa el número de días desde el 1 de enero de 4712 A.C. hasta la fecha especificada en el argumento.
"MICROSECOND" en la página 394	Devuelve la parte correspondiente a las milésimas de segundo de un valor.
"MIDNIGHT_SECONDS" en la página 395	Devuelve un valor entero que representa el número de segundos entre medianoche y un valor de hora especificado.
"MINUTE" en la página 396	Devuelve la parte del minuto de un valor.
"MONTH" en la página 398	Devuelve la parte del mes de un valor.
"MONTHNAME" en la página 399	Devuelve una serie que combina caracteres en mayúsculas y minúsculas que contiene el nombre del mes (por ejemplo, enero) de la parte correspondiente al mes del argumento, según el entorno local en el momento en que se inició la base de datos.
"QUARTER" en la página 406	Devuelve un entero que representa el trimestre del año en el que reside una fecha.
"SECOND" en la página 423	Devuelve la segunda parte de un valor.
"TIMESTAMP_FORMAT" en la página 441	Devuelve una indicación de fecha y hora a partir de una serie de caracteres (*argumento1*) que se ha interpretado utilizando una plantilla de formato (*argumento2*).
"TIMESTAMP_ISO" en la página 443	Devuelve un valor de indicación de fecha y hora basado en un argumento de fecha, de hora o de indicación de fecha y hora. Si el argumento es una fecha, inserta cero para todos los elementos de hora. Si el argumento es una hora, inserta el valor de CURRENT DATE para los elementos de fecha y ceros para el elemento de fracción de hora.
"TIMESTAMPDIFF" en la página 444	Devuelve un número estimado de intervalos de tipo *argumento1* basado en la diferencia entre dos indicaciones de fecha y hora. El segundo argumento es el resultado de restar dos tipos de indicaciones de fecha y hora y de convertir el resultado en CHAR.
"TO_CHAR" en la página 446	Devuelve una representación CHARACTER de una indicación de fecha y hora.
"TO_DATE" en la página 447	Devuelve una indicación de fecha y hora a partir de una serie de caracteres.
"VARCHAR_FORMAT" en la página 458	Devuelve una representación CHARACTER de una indicación de fecha y hora (*argumento1*) con el formato indicado en una plantilla *argumento2*).
"WEEK" en la página 462	Devuelve la semana del año a partir de un valor, donde la semana empieza el domingo.
"WEEK_ISO" en la página 463	Devuelve la semana del año a partir de un valor, donde la semana empieza el lunes.

Tabla 20. Funciones escalares de fecha y hora (continuación)

Función	Descripción
"YEAR" en la página 464	Devuelve la parte del año de un valor.

Tabla 21. Funciones escalares de particionamiento

Función	Descripción
"DBPARTITIONNUM" en la página 324	Devuelve el número de partición de base de datos de la fila. El argumento es un nombre de columna dentro de una tabla.
"HASHEDVALUE" en la página 368	Devuelve el índice de correlación de particionamiento (de 0 a 4095) de la fila. El argumento es un nombre de columna dentro de una tabla.

Tabla 22. Funciones escalares numéricas

Función	Descripción
"ABS o ABSVAL" en la página 293	Devuelve el valor absoluto de un número.
"ACOS" en la página 294	Devuelve el coarcocoseno de un número, en radianes.
"ASIN" en la página 296	Devuelve el arcoseno de un número, en radianes.
"ATAN" en la página 297	Devuelve la arcotangente de un número, en radianes.
"ATANH ATANH" en la página 299	Devuelve la arcotangente hiperbólica de un número, en radianes.
"ATAN2" en la página 298	Devuelve la arcotangente de las coordenadas x e y como un ángulo expresado en radianes.
"CEILING o CEIL" en la página 303	Devuelve el valor del entero más pequeño que es mayor o igual que un número.
"COS" en la página 313	Devuelve el coseno de un número.
"COSH" en la página 314	Devuelve el coseno hiperbólico de un número.
"COT" en la página 315	Devuelve la cotangente del argumento, donde el argumento es un ángulo expresado en radianes.
"DEGREES" en la página 332	Devuelve el número de grados de un ángulo.
"DIGITS" en la página 335	Devuelve la representación en el formato de una serie de caracteres del valor absoluto de un número.
"EXP" en la página 360	Devuelve un valor que es la base del logaritmo natural (e) elevada a la potencia especificada por el argumento.
"FLOOR" en la página 362	Devuelve el valor del entero más grande que es menor o igual que un número.
"LN" en la página 386	Devuelve el logaritmo natural de un número.
"LOG" en la página 388	Devuelve el logaritmo natural de un número (igual que LN).
"LOG10" en la página 389	Devuelve el logaritmo común (en base 10) de un número.
"MOD" en la página 397	Devuelve el resto del primer argumento dividido por el segundo argumento.
"MULTIPLY_ALT" en la página 400	Devuelve el producto de dos argumentos como un valor decimal. Esta función resulta útil cuando la suma de las precisiones del argumento es mayor que 31.

Tabla 22. Funciones escalares numéricas (continuación)

Función	Descripción
“POWER” en la página 405	Devuelve el resultado de elevar el primer argumento a la potencia del segundo argumento.
“RADIANS” en la página 407	Devuelve el número de radianes de un argumento que se expresa en grados.
“RAND” en la página 409	Devuelve un número aleatorio.
“ROUND” en la página 419	Devuelve un valor numérico que se ha redondeado el número de posiciones decimales especificado.
“SIGN” en la página 424	Devuelve el signo de un número.
“SIN” en la página 425	Devuelve el seno de un número.
“SINH SINH” en la página 426	Devuelve el seno hiperbólico de un número.
“SQRT” en la página 430	Devuelve la raíz cuadrada de un número.
“TAN” en la página 437	Devuelve la tangente de un número.
“TANH” en la página 438	Devuelve la tangente hiperbólica de un número.
“TRUNCATE o TRUNC” en la página 450	Devuelve un valor numérico que se ha truncado en el número de posiciones decimales especificado.

Tabla 23. Funciones escalares de series

Función	Descripción
“ASCII” en la página 295	Devuelve el valor en código ASCII del carácter que hay más a la izquierda del argumento como un entero.
“CHR” en la página 309	Devuelve el carácter que tiene el valor de código ASCII especificado por el argumento.
“CONCAT” en la página 312	Devuelve una serie que es la concatenación de dos series.
“DECRYPT_BIN y DECRYPT_CHAR” en la página 330	Devuelve un valor que es el resultado de descifrar datos cifrados utilizando una serie de contraseña.
“DECRYPT_BIN y DECRYPT_CHAR” en la página 330	Devuelve un valor que es el resultado de descifrar datos cifrados utilizando una serie de contraseña.
“DIFFERENCE” en la página 334	Devuelve la diferencia entre el sonido de las palabras en dos series de argumento según se determine mediante la función SOUNDEX. El valor 4 significa que las series suenan igual.
“ENCRYPT” en la página 356	Devuelve un valor que es el resultado de cifrar una expresión de serie de datos.
“GENERATE_UNIQUE” en la página 364	Devuelve una serie de caracteres de datos de bits que es exclusivo comparado con cualquier otra ejecución de la misma función.
“GETHINT” en la página 363	Devuelve la contraseña sugerida, si se encuentra una.
“INSERT” en la página 378	Devuelve una serie en la que *argumento3* bytes se han suprimido de *argumento1* (comenzando por *argumento2*) y en la que *argumento4* bytes se han insertado en *argumento1* (comenzando por *argumento2*).
“LCASE o LOWER” en la página 382	Devuelve una serie en la que todos los caracteres se han convertido a caracteres en minúsculas.

Tabla 23. Funciones escalares de series (continuación)

Función	Descripción
"LEFT" en la página 384	Devuelve los caracteres situados más a la izquierda de una serie.
"LOCATE" en la página 387	Devuelve la posición inicial de una serie contenida en otra serie.
"LTRIM" en la página 392	Elimina los blancos del principio de una expresión de serie.
"POSSTR" en la página 403	Devuelve la posición inicial de una serie contenida en otra serie.
"REPEAT" en la página 416	Devuelve una serie de caracteres compuesta por *argumento1* repetido *argumento2* veces.
"REPLACE" en la página 417	Sustituye todas las ocurrencias de *argumento2* en *argumento1* por *argumento3*.
"RIGHT" en la página 418	Devuelve los caracteres situados más a la derecha de una serie.
"RTRIM" en la página 421	Elimina los blancos del final de una expresión de serie.
"SOUNDEX" en la página 428	Devuelve un código de 4 caracteres que representa el sonido de las palabras del argumento. Este resultado se puede comparar con el sonido de otras series.
"SPACE" en la página 429	Devuelve una serie de caracteres formada por el número de blancos especificado.
"SUBSTR" en la página 431	Devuelve una subserie de una serie.
"TRANSLATE" en la página 448	Devuelve una serie en la que uno o más caracteres de una serie se han convertido en otros caracteres.
"UCASE o UPPER" en la página 454	Devuelve una serie en la que todos los caracteres se han convertido a caracteres en mayúsculas.

Tabla 24. Funciones escalares diversas

Función	Descripción
"COALESCE" en la página 311	Devuelve el primer argumento que no es nulo.
"DEREF" en la página 333	Devuelve una instancia del tipo de destino del argumento del tipo de referencia.
"EVENT_MON_STATE" en la página 359	Devuelve el estado operativo de un determinado supervisor de sucesos.
"HEX" en la página 370	Devuelve una representación hexadecimal de un valor.
"IDENTITY_VAL_LOCAL" en la página 373	Devuelve el valor asignado más reciente correspondiente a una columna de entidad.
"LENGTH" en la página 385	Devuelve la longitud de un valor.
"NULLIF" en la página 402	Devuelve un valor nulo si los argumentos son iguales; de lo contrario, devuelve el valor del primer argumento.
"RAISE_ERROR" en la página 408	Emite un error a la SQLCA. El sqlstate que se devolverá se indica mediante *argumento1*. El segundo argumento contiene el texto que debe devolverse.
"REC2XML" en la página 411	Devuelve una serie formateada codificada en XML que contiene nombres de columna y datos de columna.

Tabla 24. Funciones escalares diversas (continuación)

Función	Descripción
"TABLE_NAME" en la página 434	Devuelve un nombre no calificado de una tabla o vista, basado en el nombre de objeto especificado en *argumento1* y en el nombre de esquema opcional especificado en *argumento2*. El valor devuelto se utiliza para resolver los alias.
"TABLE_SCHEMA" en la página 435	Devuelve la parte correspondiente al nombre de esquema de un nombre de tabla o de vista de dos partes (especificado por el nombre del objeto en *argumento1* y por el nombre de esquema opcional en *argumento2*). El valor devuelto se utiliza para resolver los alias.
"TYPE_ID" en la página 451	Devuelve el identificador interno de tipo de datos del tipo de datos dinámico del argumento. El resultado de esta función no se puede transportar entre bases de datos.
"TYPE_NAME" en la página 452	Devuelve el nombre no calificado del tipo de datos dinámico del argumento.
"TYPE_SCHEMA" en la página 453	Devuelve el nombre del esquema del tipo de datos dinámico del argumento.
"VALUE" en la página 455	Devuelve el primer argumento que no es nulo.

Funciones agregadas

El argumento de una función de columna es un conjunto de valores derivados de una expresión. La expresión puede incluir columnas, pero no puede incluir una *selecciónconpleta-escalar* ni otra función de columna (SQLSTATE 42607). El ámbito del conjunto es un grupo o una tabla resultante intermedia.

Si se especifica una cláusula GROUP BY en una consulta y el resultado intermedio de las cláusulas FROM, WHERE, GROUP BY y HAVING es el conjunto vacío, las funciones de columna no se aplican; el resultado de la consulta es el conjunto vacío; el SQLCODE se establece en +100 y el SQLSTATE se establece en '02000'.

Si *no* se especifica una cláusula GROUP BY en una consulta y el resultado intermedio de las cláusulas FROM, WHERE y HAVING es el conjunto vacío, las funciones de columna se aplican al conjunto vacío.

Por ejemplo, el resultado de la siguiente sentencia SELECT es el número de valores diferenciado de JOBCODE para los empleados en el departamento D01:

```
SELECT COUNT(DISTINCT JOBCODE)
  FROM CORPDATA.EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

La palabra clave DISTINCT no se considera un argumento de la función, sino una especificación de una operación que se realiza antes de aplicar la función. Si se especifica DISTINCT, se eliminan los valores duplicados. Si se especifica ALL implícita o explícitamente, no se eliminan los valores duplicados.

Se pueden utilizar expresiones en las funciones de columna. Por ejemplo:

```
SELECT MAX(BONUS + 1000)
  INTO :TOP_SALESREP_BONUS
  FROM EMPLOYEE
  WHERE COMM > 5000
```

Las funciones de columna puede esta calificadas mediante un nombre de esquema (por ejemplo, SYSIBM.COUNT(*)).

Información relacionada:

• "Consultas de SQL" en la página 469

AVG

El esquema es SYSIBM.

La función AVG devuelve el promedio de un conjunto de números.

Los valores del argumento deben ser números (sólo tipos internos) y su suma debe estar dentro del rango del tipo de datos del resultado, excepto para un tipo de datos de resultado decimal. Para los resultados decimales, la suma debe estar dentro del rango soportado por un tipo de datos decimal que tenga una precisión de 31 y una escala idéntica a la escala de los valores del argumento. El resultado puede ser nulo.

El tipo de datos del resultado es el mismo que el tipo de datos de los valores del argumento, excepto que:

- El resultado es un entero grande si los valores del argumento son enteros pequeños.
- El resultado es de coma flotante de precisión doble si los valores del argumento son de coma flotante de precisión simple.

Si el tipo de datos de los valores del argumento es decimal con la precisión *p* y la escala *s*, la precisión del resultado es 31 y la escala es 31-*p*+*s*.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, se eliminan los valores duplicados redundantes.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es el valor promedio del conjunto.

El orden en el que los valores se añaden es indefinido, pero cada resultado intermedio debe estar en el rango del tipo de datos del resultado.

Si el tipo del resultado es entero, se pierde la parte correspondiente a la fracción del promedio.

Ejemplos:

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal AVERAGE (decimal(5,2)) en el nivel promedio de los trabajadores (PRSTAFF) de los proyectos del departamento (DEPTNO) 'D11'.

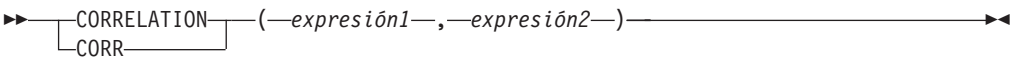
  ```
  SELECT AVG(PRSTAFF)
    INTO :AVERAGE
    FROM PROJECT
    WHERE DEPTNO = 'D11'
  ```

  Da como resultado que AVERAGE se establece en 4,25 (es decir 17/4) cuando se utiliza la tabla de ejemplo.
- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal ANY_CALC (decimal(5,2)) en el promedio de cada valor de nivel exclusivo de los trabajadores (PRSTAFF) de proyectos del departamento (DEPTNO) 'D11'.

```
SELECT AVG(DISTINCT PRSTAFF)
  INTO :ANY_CALC
  FROM PROJECT
  WHERE DEPTNO = 'D11'
```

El resultado es que ANY_CALC se establece en 4,66 (es decir 14/3) cuando se utiliza la tabla de ejemplo.

CORRELATION



El esquema es SYSIBM.

La función CORRELATION devuelve el coeficiente de correlación de un conjunto de pares de números.

Los valores del argumento deben ser números.

El tipo de datos del resultado es de coma flotante de precisión doble. El resultado puede ser nulo. Cuando no es nulo, el resultado está entre –1 y 1.

La función se aplica al conjunto de pares (*expresión1*, *expresión2*) derivado de los valores del argumento por la eliminación de todos los pares para los que *expresión1* o *expresión2* es nulo.

Si la función se aplica a un conjunto vacío o si STDDEV(*expresión1*) o STDDEV(*expresión2*) es igual a cero, el resultado es un valor nulo. De lo contrario, el resultado es el coeficiente de correlación para los pares de valores del conjunto. El resultado es equivalente a la expresión siguiente:

```
COVARIANCE(expresión1,expresión2)/
(STDDEV(expresión1)*
STDDEV(expresión2))
```

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

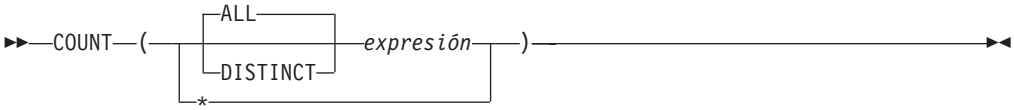
Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal CORRLN (coma flotante de precisión doble) en la correlación entre salario y bonificación para los empleados del departamento (WORKDEPT) 'A00'.

  ```
  SELECT CORRELATION(SALARY, BONUS)
    INTO :CORRLN
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  CORRLN se establece en 9,99853953399538E-001 aproximadamente cuando se utiliza la tabla de ejemplo.

COUNT



El esquema es SYSIBM.

La función COUNT devuelve el número de filas o valores de un conjunto de filas o valores.

El tipo de datos de *expresión* no puede ser un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado para cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907).

El resultado de la función es un entero grande. El resultado no puede ser nulo.

El argumento de COUNT(*) es un conjunto de filas. El resultado es el número de filas del conjunto. Una fila que sólo incluye valores NULL se incluye en la cuenta.

El argumento de COUNT(DISTINCT *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos y duplicados. El resultado es el número de distintos valores no nulos del conjunto.

El argumento de COUNT(*expresión*) o COUNT(ALL *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. El resultado es el número de valores no nulos del conjunto, incluyendo los duplicados.

Ejemplos:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal FEMALE (int) en el número de filas en que el valor de la columna SEX es 'F'.

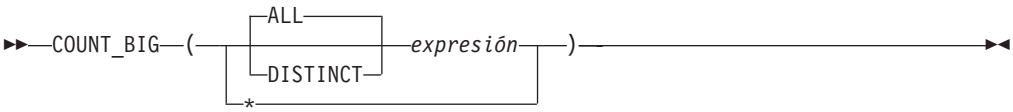
  ```
  SELECT COUNT(*)
    INTO :FEMALE
    FROM EMPLOYEE
    WHERE SEX = 'F'
  ```

  El resultado es que FEMALE se establece en 13 cuando se utiliza la tabla de ejemplo.
- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal FEMALE_IN_DEPT (int) en el número de departamentos (WORKDEPT) que tienen como mínimo una mujer como miembro.

  ```
  SELECT COUNT(DISTINCT WORKDEPT)
    INTO :FEMALE_IN_DEPT
    FROM EMPLOYEE
    WHERE SEX = 'F'
  ```

  El resultado es que FEMALE_IN_DEPT se establece en 5 cuando se utiliza la tabla de ejemplo. (Hay como mínimo una mujer en los departamentos A00, C01, D11, D21 y E11.)

COUNT_BIG



El esquema es SYSIBM.

La función COUNT_BIG devuelve el número de filas o valores de un conjunto de filas o valores. Es similar a COUNT excepto que el resultado puede ser mayor que el valor máximo de entero.

El tipo de datos resultante de *expresión* no puede ser un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado para cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907).

El resultado de la función es un decimal con la precisión 31 y la escala 0. El resultado no puede ser nulo.

El argumento de COUNT_BIG(*) es un conjunto de filas. El resultado es el número de filas del conjunto. Una fila que sólo incluye valores NULL se incluye en la cuenta.

El argumento de COUNT_BIG(DISTINCT *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos y duplicados. El resultado es el número de distintos valores no nulos del conjunto.

El argumento de COUNT_BIG(*expresión*) o COUNT_BIG(ALL *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. El resultado es el número de valores no nulos del conjunto, incluyendo los duplicados.

Ejemplos:

- Consulte los ejemplos de COUNT y sustituya COUNT_BIG por las apariciones de COUNT. Los resultados son los mismos excepto por el tipo de datos del resultado.
- Algunas aplicaciones pueden necesitar la utilización de COUNT pero necesitan dar soporte a valores mayores que el entero más grande. Esto se puede conseguir mediante la utilización de funciones derivadas definidas por el usuario y la definición de la vía de acceso de SQL. Las siguientes series de sentencias muestran cómo crear una función derivada para dar soporte a COUNT(*) basándose en COUNT_BIG y devolver un valor decimal con una precisión de 15. La vía de acceso de SQL se establece de manera que se utilice la función derivada basada en COUNT_BIG en las sentencias subsiguientes, tal como la consulta mostrada.

  ```
  CREATE FUNCTION RICK.COUNT() RETURNS DECIMAL(15,0)
    SOURCE SYSIBM.COUNT_BIG();
  SET CURRENT FUNCTION PATH RICK, SYSTEM PATH;
  SELECT COUNT(*) FROM EMPLOYEE;
  ```

  Observe que la función derivada se define sin parámetros para dar soporte a COUNT(*). Esto sólo es efectivo si utiliza COUNT como nombre de la función y no califica la función con el nombre de esquema cuando se utiliza. Para

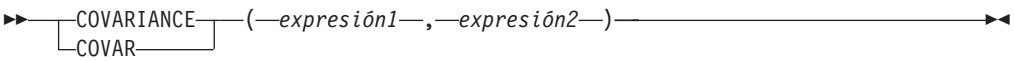
conseguir el mismo efecto que COUNT(*) con un nombre distinto de COUNT, invoque la función sin parámetros. Por lo tanto, si RICK.COUNT se ha definido como RICK.MYCOUNT, la consulta se tendría que haber escrito de la siguiente manera:

```
SELECT MYCOUNT() FROM EMPLOYEE;
```

Si la cuenta se efectúa en una columna específica, la función derivada debe especificar el tipo de columna. Las sentencias siguientes crean una función derivada que tomará cualquier columna CHAR como argumento y utilizará COUNT_BIG para realizar el contaje.

```
CREATE FUNCTION RICK.COUNT(CHAR()) RETURNS DOUBLE
  SOURCE SYSIBM.COUNT_BIG(CHAR());
SELECT COUNT(DISTINCT WORKDEPT) FROM EMPLOYEE;
```

COVARIANCE



El esquema es SYSIBM.

La función COVARIANCE devuelve la covarianza (del contenido) de un conjunto de pares de números.

Los valores del argumento deben ser números.

El tipo de datos del resultado es de coma flotante de precisión doble. El resultado puede ser nulo.

La función se aplica al conjunto de pares (*expresión1*, *expresión2*) derivado de los valores del argumento por la eliminación de todos los pares para los que *expresión1* o *expresión2* es nulo.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la covarianza de los pares de valores del conjunto. El resultado es equivalente a lo siguiente:

1. Establezca que avgexp1 es el resultado de AVG(*expresión1*) y que avgexp2 es el resultado de AVG(*expresión2*).
2. El resultado de COVARIANCE(*expresión1*, *expresión2*) es AVG((*expresión1* - avgexp1) * (*expresión2* - avgexp2)

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal COVARNCE (coma flotante de precisión doble) en la covarianza entre salario y bonificación para los empleados del departamento (WORKDEPT) 'A00'.

```
SELECT COVARIANCE(SALARY, BONUS)
  INTO :COVARNCE
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00'
```

  COVARNCE se establece en 1,68888888888889E+006 aproximadamente cuando se utiliza la tabla de ejemplo.

GROUPING

El esquema es SYSIBM.

Utilizada con conjuntos-agrupaciones y supergrupos, la función GROUPING devuelve un valor que indica si una fila devuelta en un conjunto de respuestas de GROUP BY es una fila generada por un conjunto de agrupaciones que excluye la columna representada por la *expresión* o no.

El argumento puede ser de cualquier tipo, pero debe ser un elemento de una cláusula GROUP BY.

El resultado de la función es un entero pequeño. Se establece en uno de los valores siguientes:

1 El valor de la *expresión* de la fila devuelta es un valor nulo y la fila se ha generado por el supergrupo. Esta fila generada puede utilizarse para proporcionar valores de subtotales para la expresión GROUP BY.

0 El valor no es el de arriba.

Ejemplo:

La siguiente consulta:

```
SELECT SALES_DATE, SALES_PERSON,
    SUM(SALES) AS UNITS_SOLD,
    GROUPING(SALES_DATE) AS DATE_GROUP,
    GROUPING(SALES_PERSON) AS SALES_GROUP
  FROM SALES
  GROUP BY CUBE (SALES_DATE, SALES_PERSON)
  ORDER BY SALES_DATE, SALES_PERSON
```

da como resultado:

```
SALES_DATE SALES_PERSON    UNITS_SOLD  DATE_GROUP  SALES_GROUP
---------- --------------- ----------- ----------- -----------
12/31/1995 GOUNOT                    1           0           0
12/31/1995 LEE                       6           0           0
12/31/1995 LUCCHESSI                 1           0           0
12/31/1995 -                         8           0           1
03/29/1996 GOUNOT                   11           0           0
03/29/1996 LEE                      12           0           0
03/29/1996 LUCCHESSI                 4           0           0
03/29/1996 -                        27           0           1
03/30/1996 GOUNOT                   21           0           0
03/30/1996 LEE                      21           0           0
03/30/1996 LUCCHESSI                 4           0           0
03/30/1996 -                        46           0           1
03/31/1996 GOUNOT                    3           0           0
03/31/1996 LEE                      27           0           0
03/31/1996 LUCCHESSI                 1           0           0
03/31/1996 -                        31           0           1
04/01/1996 GOUNOT                   14           0           0
04/01/1996 LEE                      25           0           0
04/01/1996 LUCCHESSI                 4           0           0
04/01/1996 -                        43           0           1
-          GOUNOT                   50           1           0
```

```
-          LEE                    91          1          0
-          LUCCHESSI              14          1          0
-          -                     155          1          1
```

Una aplicación puede reconocer una fila de subtotales de SALES_DATE por el hecho de que el valor de DATE_GROUP es 0 y el valor de SALES_GROUP es 1. Una fila de subtotales SALES_PERSON puede reconocerse por el hecho de que el valor de DATE_GROUP es 1 y el valor de SALES_GROUP es 0. Un fila de total general puede reconocerse por el valor 1 de DATE_GROUP y SALES_GROUP.

Información relacionada:

• "Subselección" en la página 470

MAX

El esquema es SYSIBM.

La función MAX devuelve el valor máximo de un conjunto de valores.

Los valores del argumento pueden ser de cualquier tipo interno que no sea una serie larga ni DATALINK.

El tipo de datos resultante de *expresión* no puede ser un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado para cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907).

El tipo de datos, la longitud y la página de códigos del resultado son iguales que el tipo de datos, la longitud y la página de códigos de los valores del argumento. El resultado se considera un valor derivado y puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es el valor máximo del conjunto.

La especificación de DISTINCT no tiene ningún efecto en el resultado y, por lo tanto, no es aconsejable. Se incluye para la compatibilidad con otros sistemas relacionados.

Ejemplos:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal MAX_SALARY (decimal(7,2)) en el valor del salario máximo mensual (SALARY/12).

  ```
  SELECT MAX(SALARY) / 12
    INTO :MAX_SALARY
    FROM EMPLOYEE
  ```

  El resultado es que MAX_SALARY se establece en 4395,83 cuando se utiliza esta tabla de ejemplo.
- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal LAST_PROJ(char(24)) en el nombre de proyecto (PROJNAME) que es el último en el orden de clasificación.

  ```
  SELECT MAX(PROJNAME)
    INTO :LAST_PROJ
    FROM PROJECT
  ```

  Da como resultado que LAST_PROJ se establece en 'WELD LINE PLANNING' cuando se utiliza la tabla de ejemplo.
- De manera parecida al ejemplo anterior, establezca la variable del lenguaje principal LAST_PROJ (char(40)) en el nombre del proyecto que es el último en el

orden de clasificación cuando se concatena un nombre de proyecto con la variable del lenguaje principal PROJSUPP. PROJSUPP es '_Support'; tiene un tipo de datos char(8).

```
SELECT MAX(PROJNAME CONCAT PROJSUPP)
  INTO :LAST_PROJ
  FROM PROJECT
```

Da como resultado que LAST_PROJ se establece en 'WELD LINE PLANNING_SUPPORT' cuando se utiliza la tabla de ejemplo.

MIN

La función MIN devuelve el valor mínimo de un conjunto de valores.

Los valores del argumento pueden ser de cualquier tipo interno que no sea una serie larga ni DATALINK.

El tipo de datos resultante de *expresión* no puede ser un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado para cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907).

El tipo de datos, la longitud y la página de códigos del resultado son iguales que el tipo de datos, la longitud y la página de códigos de los valores del argumento. El resultado se considera un valor derivado y puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos.

Si la función se aplica a un conjunto vacío, el resultado de la función es un valor nulo. De lo contrario, el resultado es el valor mínimo del conjunto.

La especificación de DISTINCT no tiene ningún efecto en el resultado y, por lo tanto, no es aconsejable. Se incluye para la compatibilidad con otros sistemas relacionados.

Ejemplos:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal COMM_SPREAD (decimal(7,2)) en la diferencia entre la comisión máxima y mínima (COMM) de los miembros del departamento (WORKDEPT) 'D11'.

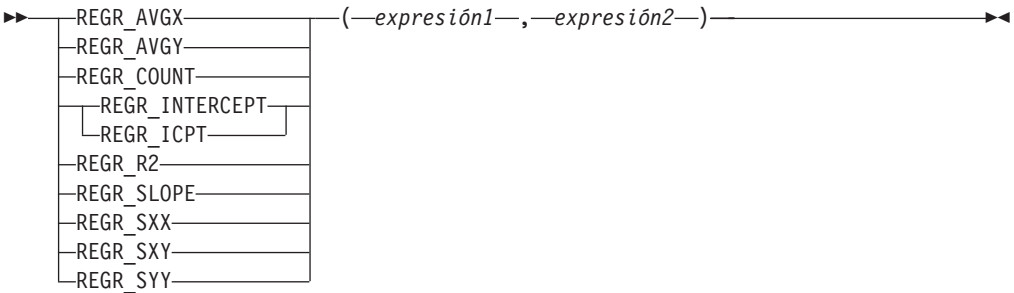
  ```
  SELECT MAX(COMM) - MIN(COMM)
    INTO :COMM_SPREAD
    FROM EMPLOYEE
    WHERE WORKDEPT = 'D11'
  ```

  El resultado es que COMM_SPREAD se establece en 1118 (es decir, 2580 - 1462) cuando se utiliza la tabla de ejemplo.
- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal (FIRST_FINISHED (char(10)) en la fecha de finalización estimada (PRENDATE) del primer proyecto que se ha de terminar.

  ```
  SELECT MIN(PRENDATE)
    INTO :FIRST_FINISHED
    FROM PROJECT
  ```

  Da como resultado que FIRST_FINISHED se establece en '1982-09-15' cuando se utiliza la tabla de ejemplo.

Funciones de regresión



El esquema es SYSIBM.

Las funciones de regresión soportan la adecuación de una línea de regresión mínimo-cuadrados-normales del formato y = a * x + b para un conjunto de pares de números. El primer elemento de cada par (*expresión1*) se interpreta como un valor de la variable dependiente (es decir, un ″valor y″). El segundo elemento de cada par (*expresión2*) se interpreta como un valor de la variable independiente (es decir, un ″valor x″).

La función REGR_COUNT devuelve el número de pares de números no nulos utilizados para acomodar la línea de regresión (vea más abajo).

La función REGR_INTERCEPT (o REGR_ICPT) devuelve la intersección y de la línea de regresión (″b″ en la ecuación anterior).

La función REGR_R2 devuelve el coeficiente de determinación (″cuadrado-R″ o ″mejor-adecuación″) para la regresión.

La función REGR_SLOPE devuelve la inclinación de la línea (″a″ en la ecuación anterior).

Las funciones REGR_AVGX, REGR_AVGY, REGR_SXX, REGR_SXY y REGR_SYY devuelven cantidades que pueden utilizarse para calcular varias estadísticas de diagnóstico necesarias para la evaluación de la calidad y la validez estadística del modelo de regresión (vea más abajo).

Los valores del argumento deben ser números.

El tipo de datos del resultado de REGR_COUNT es un entero. Para las demás funciones, el tipo de datos del resultado es de coma flotante de precisión doble. El resultado puede ser nulo. Cuando no es nulo, el resultado de REGR_R2 está comprendido entre 0 y 1 y el resultado de REGR_SXX y REGR_SYY no es negativo.

Cada función se aplica al conjunto de pares (*expresión1*, *expresión2*) derivado de los valores del argumento por la eliminación de todos los pares para los que *expresión1* o *expresión2* es nulo.

Si el conjunto no está vacío y VARIANCE(*expresión2*) es positivo, REGR_COUNT devuelve el número de pares no nulos del conjunto y las demás funciones devuelven los resultados que se definen de la siguiente manera:

REGR_SLOPE(*expresión1*,*expresión2*) =
COVARIANCE(*expresión1*,*expresión2*)/**VARIANCE**(*expresión2*)

```
REGR_INTERCEPT(expresión1, expresión2) =
AVG(expresión1) - REGR_SLOPE(expresión1, expresión2) * AVG(expresión2)
REGR_R2(expresión1, expresión2) =
POWER(CORRELATION(expresión1, expresión2), 2) if VARIANCE(expresión1)>0
REGR_R2(expresión1, expresión2) = 1 if VARIANCE(expresion1)=0
REGR_AVGX(expresión1, expresión2) = AVG(expresión2)
REGR_AVGY(expresión1, expresión2) = AVG(expresión1)
REGR_SXX(expresión1, expresión2) =
REGR_COUNT(expresión1, expresión2) * VARIANCE(expresión2)
REGR_SYY(expresión1, expresión2) =
REGR_COUNT(expresión1, expresión2) * VARIANCE(expresión1)
REGR_SXY(expresión1, expresión2) =
REGR_COUNT(expresión1, expresión2) * COVARIANCE(expresión1, expresión2)
```

Si el conjunto no está vacío y VARIANCE(*expresión2*) es igual a cero, la línea de regresión tiene una inclinación infinita o no está definida. En este caso, las funciones REGR_SLOPE, REGR_INTERCEPT y REGR_R2 devuelven cada una un valor nulo y las demás funciones devuelven valores tal como se ha definido arriba. Si el conjunto está vacío, REGR_COUNT devuelve cero y las demás funciones devuelven un valor nulo.

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

Las funciones de regresión se calculan simultáneamente durante un solo paso a través de los datos. En general, es más eficaz utilizar las funciones de regresión para calcular las estadísticas necesarias para un análisis de regresión que realizar cálculos equivalentes utilizando las funciones normales de columna como AVERAGE, VARIANCE, COVARIANCE, etcétera.

Las estadísticas de diagnóstico normales que acompañan a un análisis de regresión-lineal se pueden calcular en términos de las funciones anteriores. Por ejemplo:

R2 ajustada
1 - ((1 - REGR_R2) * ((REGR_COUNT - 1) / (REGR_COUNT - 2)))

Error estándar
SQRT((REGR_SYY-
(POWER(REGR_SXY,2)/REGR_SXX))/(REGR_COUNT-2))

Suma total de cuadrados
REGR_SYY

Suma de cuadrados de regresión
POWER(REGR_SXY,2) / REGR_SXX

Suma de cuadrados residuales
(Suma total de cuadrados)-(Suma de cuadrados de regresión)

t estadística de inclinación
REGR_SLOPE * SQRT(REGR_SXX) / (Error estándar)

t estadística para intersección y
REGR_INTERCEPT/((Error estándar)*SQRT((1/REGR_COUNT)+
(POWER(REGR_AVGX,2)/REGR_SXX))

Ejemplo:

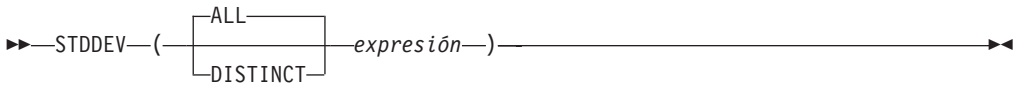
- Utilizando la tabla EMPLOYEE, calcule la línea de regresión de cuadrados-mínimos-normales que expresa la bonificación de un empleado del departamento (WORKDEPT) 'A00' como una función lineal del salario del empleado. Establezca las variables del lenguaje principal SLOPE, ICPT, RSQR (coma flotante de precisión doble) en la inclinación, intersección y coeficiente de determinación de la línea de regresión, respectivamente. Establezca también las variables del lenguaje principal AVGSAL y AVGBONUS en el salario medio y la bonificación media, respectivamente, de los empleados del departamento 'A00', y establezca la variable del lenguaje principal CNT (entero) en el número de empleados del departamento 'A00' para los que están disponibles los datos de salario y de bonificación. Almacene las demás estadísticas de regresión en las variables del lenguaje principal SXX, SYY y SXY.

```
SELECT REGR_SLOPE(BONUS,SALARY), REGR_INTERCEPT(BONUS,SALARY),
REGR_R2(BONUS,SALARY), REGR_COUNT(BONUS,SALARY),
REGR_AVGX(BONUS,SALARY), REGR_AVGY(BONUS,SALARY),
REGR_SXX(BONUS,SALARY), REGR_SYY(BONUS,SALARY),
REGR_SXY(BONUS,SALARY)
INTO :SLOPE, :ICPT,
          :RSQR, :CNT,
          :AVGSAL, :AVGBONUS,
          :SXX, :SYY,
          :SXY
FROM EMPLOYEE
WHERE WORKDEPT = 'A00'
```

Al utilizar la tabla de ejemplo, las variables del lenguaje principal se establecen en los siguientes valores aproximados:

```
SLOPE:  +1.71002671916749E-002
ICPT:  +1.00871888623260E+002
RSQR:  +9.99707928128685E-001
CNT:  3
AVGSAL:  +4.28333333333333E+004
AVGBONUS:  +8.33333333333333E+002
SXX:  +2.96291666666667E+008
SYY:  +8.66666666666667E+004
SXY:  +5.06666666666667E+006
```

STDDEV



El esquema es SYSIBM.

La función STDDEV devuelve la desviación estándar de un conjunto de números.

Los valores del argumento deben ser números.

El tipo de datos del resultado es de coma flotante de precisión doble. El resultado puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, se eliminan los valores duplicados redundantes.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la desviación estándar de los valores del conjunto.

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal DEV (coma flotante de precisión doble) en la desviación estándar de los salarios para los empleados del departamento (WORKDEPT) 'A00'.

  ```
  SELECT STDDEV(SALARY)
    INTO :DEV
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  Da como resultado que DEV se establece en 9938,00 aproximadamente cuando se utiliza la tabla de ejemplo.

SUM

El esquema es SYSIBM.

La función SUM devuelve la suma de un conjunto de números.

Los valores del argumento deben ser números (sólo tipos internos) y su suma debe estar dentro del rango del tipo de datos del resultado.

El tipo de datos del resultado es el mismo que el tipo de datos de los valores del argumento excepto que:

- El resultado es un entero grande si los valores del argumento son enteros pequeños.
- El resultado es de coma flotante de precisión doble si los valores del argumento son de coma flotante de precisión simple.

Si el tipo de datos de los valores del argumento es decimal, la precisión del resultado es 31 y la escala es la misma que la de los valores del argumento. El resultado puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, también se eliminan los valores duplicados redundantes.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la suma de los valores del conjunto.

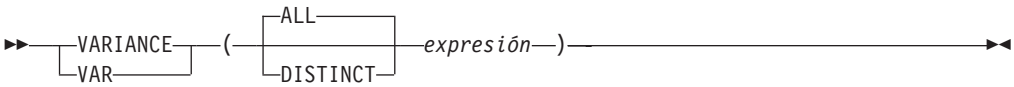
Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal JOB_BONUS (decimal(9,2)) en el total de bonificaciones (BONUS) pagadas a los conserjes (JOB='CLERK').

  ```
  SELECT SUM(BONUS)
    INTO :JOB_BONUS
    FROM EMPLOYEE
    WHERE JOB = 'CLERK'
  ```

  El resultado es que JOB_BONUS se establece en 2800 cuando se utiliza la tabla de ejemplo.

VARIANCE



El esquema es SYSIBM.

La función VARIANCE devuelve la varianza de un conjunto de números.

Los valores del argumento deben ser números.

El tipo de datos del resultado es de coma flotante de precisión doble. El resultado puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, se eliminan los valores duplicados redundantes.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la varianza de los valores del conjunto.

El orden en el que los valores se añaden es indefinido, pero cada resultado intermedio debe estar en el rango del tipo de datos del resultado.

Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal VARNCE (coma flotante de precisión doble) en la varianza de los salarios para los empleados del departamento (WORKDEPT) 'A00'.

  ```
  SELECT VARIANCE(SALARY)
    INTO :VARNCE
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  Da como resultado que VARNCE se establece en 98763888,88 aproximadamente cuando se utiliza la tabla de ejemplo.

Funciones escalares

Una función escalar se puede utilizar siempre que se pueda utilizar una expresión. Sin embargo, las restricciones que se aplican a la utilización de expresiones y a las funciones de columna también se aplican cuando se utiliza una expresión o una función de columna en una función escalar. Por ejemplo, el argumento de una función escalar sólo puede ser una función de columna si está permitida una función de columna en el contexto en el que se utiliza la función escalar.

Las restricciones en la utilización de funciones de columna no se aplican a las funciones escalares, porque una función escalar se aplica a un solo valor en lugar de a un conjunto de valores.

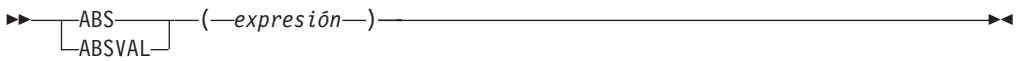
El resultado de la siguiente sentencia SELECT contiene un mismo número de filas igual al número de empleados que hay en el departamento D01:

```
SELECT EMPNO, LASTNAME, YEAR(CURRENT DATE - BRTHDATE)
  FROM EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

Las funciones escalares puede esta calificadas mediante un nombre de esquema (por ejemplo, SYSIBM.CHAR(123)).

En una base de datos Unicode, todas las funciones escalares que acepten una serie de caracteres o gráfica aceptarán todos los tipos de serie para los que se soporte la conversión.

ABS o ABSVAL



El esquema es SYSIBM.

Esta función está disponible por primera vez en el FixPak 2 de la Versión 7.1. La versión SYSFUN de la función ABS (o ABSVAL) continúa estando disponible.

Devuelve el valor absoluto del argumento. El argumento puede ser de cualquier tipo de datos numérico interno.

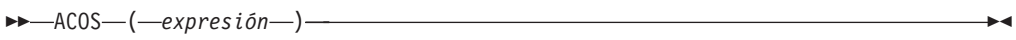
El resultado tiene el mismo tipo de datos y el mismo atributo de longitud que el argumento. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo. Si el argumento es el valor negativo máximo para SMALLINT, INTEGER o BIGINT, el resultado es un error de desbordamiento.

Ejemplo:

```
ABS(-51234)
```

devuelve un INTEGER con un valor de 51234.

ACOS



El esquema es SYSIBM. (La versión SYSFUN de la función ACOS continúa estando disponible).

Devuelve el arcocoseno del argumento como un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

Ejemplo:

Supongamos que la variable del lenguaje principal ACOSINE es una variable del lenguaje principal DECIMAL(10,9) con un valor de 0,070737202.

```
SELECT ACOS(:ACOSINE)
  FROM SYSIBM.SYSDUMMY1
```

Esta sentencia devuelve el valor aproximado 1,49.

ASCII

El esquema es SYSFUN.

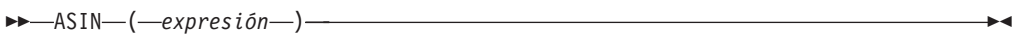
Devuelve el valor de código ASCII del carácter situado más a la izquierda del argumento como un entero.

El argumento puede ser de cualquier tipo de serie de caracteres interno. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función. Para un VARCHAR la longitud máxima es de 4.000 bytes y para un CLOB la longitud máxima es de 1.048.576 bytes. LONG VARCHAR se convierte a CLOB para que lo procese la función.

El resultado de la función siempre es INTEGER.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

ASIN



El esquema es SYSIBM. (La versión SYSFUN de la función ASIN continúa estando disponible).

Devuelve el arcoseno del argumento como un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo numérico interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

ATAN

El esquema es SYSIBM. (La versión SYSFUN de la función ATAN continúa estando disponible).

Devuelve la tangente del arco del argumento como un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

ATAN2

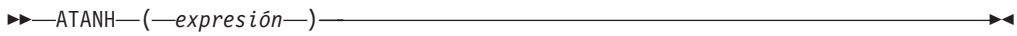
El esquema es SYSIBM. (La versión SYSFUN de la función ATAN2 continúa estando disponible).

Devuelve la tangente del arco de las coordenadas x e y como un ángulo expresado en radianes. Las coordenadas x e y se especifican por el primer y el segundo argumento, respectivamente.

El primer argumento y el segundo pueden ser de cualquier tipo de datos numérico interno. Los dos se convierten a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

ATANH ATANH



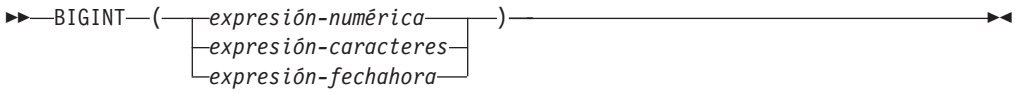
El esquema es SYSIBM.

Devuelve la arcotangente hiperbólica del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

BIGINT



El esquema es SYSIBM.

La función BIGINT devuelve una representación entera de 64 bits de un número, serie de caracteres, fecha, hora o indicación de fecha y hora en forma de una constante entera. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

Si el argumento es una *expresión-numérica*, el resultado es el mismo número que sería si el argumento se asignase a una columna o variable de enteros superiores. Si la parte correspondiente a los enteros del argumento no está dentro del rango de enteros, se produce un error. La parte correspondiente a los decimales del argumento se trunca si está presente.

expresión-caracteres
Una expresión que devuelve un valor de serie de caracteres de longitud no mayor que la longitud máxima de una constante de caracteres. Se eliminan los blancos iniciales y de cola y la serie resultante debe ajustarse a las reglas para la formación de una constante de enteros SQL (SQLSTATE 22018). La serie de caracteres no puede ser una serie larga.

Si el argumento es una *expresión-caracteres*, el resultado es el mismo número que sería si la constante de enteros correspondiente se asignase a una columna o variable de enteros superiores.

expresión-fechahora
Una expresión que sea uno de los tipos de datos siguientes:

- DATE. El resultado es un valor BIGINT que representa la fecha como *aaaammdd*.
- TIME. El resultado es un valor BIGINT que representa la hora como *hhmmss*.
- TIMESTAMP. El resultado es un valor BIGINT que representa la indicación de fecha y hora como *aaaammddhhmmss*. La parte de los microsegundos del valor de indicación de fecha y hora no se incluye en el resultado.

El resultado de la función es un entero superior. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos:

- En la tabla ORDERS_HISTORY, cuente el número de órdenes y devuelva el resultado como un valor de entero superior.

```
SELECT BIGINT (COUNT_BIG(*))
  FROM ORDERS_HISTORY
```

- Utilizando la tabla EMPLOYEE, seleccione la columna EMPNO en el formato de enteros superiores para procesarla más en la aplicación.

```
SELECT BIGINT (EMPNO) FROM EMPLOYEE
```

- Supongamos que la columna RECEIVED (indicación de fecha y hora) tiene un valor interno equivalente a '1988-12-22-14.07.21.136421'.

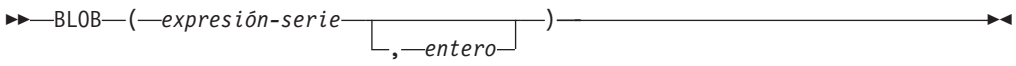
  ```
  BIGINT(RECEIVED)
  ```

  da como resultado el valor `19 881 222 140 721`.
- Supongamos que la columna STARTTIME (hora) tiene un valor interno equivalente a '12:03:04'.

  ```
  BIGINT(STARTTIME)
  ```

  da como resultado el valor `120 304`.

BLOB



El esquema es SYSIBM.

La función BLOB devuelve una representación BLOB de una serie de cualquier tipo.

expresión-serie
Una *expresión-serie* cuyo valor puede ser una serie de caracteres, una serie gráfica o una serie binaria.

entero
Un valor entero que especifica el atributo de longitud del tipo de datos BLOB resultante. Si no se especifica *entero*, el atributo de longitud del resultado es el mismo que la longitud de la entrada, excepto cuando la entrada es gráfica. En este caso, el atributo de longitud del resultado es el doble de la longitud de la entrada.

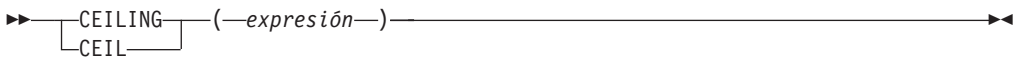
El resultado de la función es un BLOB. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos

- Suponiendo una tabla con una columna BLOB denominada TOPOGRAPHIC_MAP y una columna VARCHAR denominada MAP_NAME, localice los mapas que contienen la serie 'Pellow Island' y devuelva una sola serie binaria con el nombre del mapa concatenado delante del mapa real.

```
SELECT BLOB(MAP_NAME  || ':  ') || TOPOGRAPHIC_MAP
 FROM ONTARIO_SERIES_4
  WHERE TOPOGRAPHIC_MAP LIKE BLOB('%Pellow Island%')
```

CEILING o CEIL



El esquema es SYSIBM. (La versión SYSFUN de la función CEILING o CEIL continúa estando disponible).

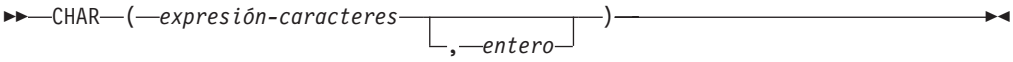
Devuelve el valor del entero más pequeño que es mayor o igual que el argumento.

El argumento puede ser de cualquier tipo numérico interno. El resultado de la función tiene el mismo tipo de datos y el mismo atributo de longitud que el argumento, con la excepción de que la escala es 0 si el argumento es DECIMAL. Por ejemplo, un argumento con un tipo de datos de DECIMAL(5,5) devuelve DECIMAL(5,0).

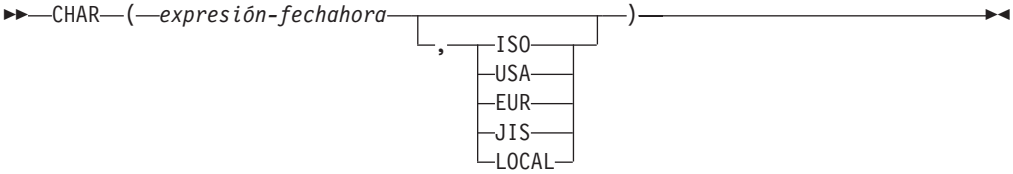
El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

CHAR

De caracteres a caracteres:

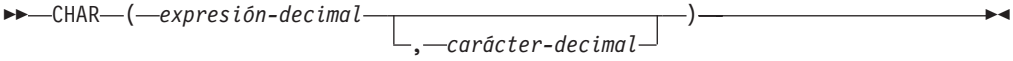


De fecha y hora a caracteres:

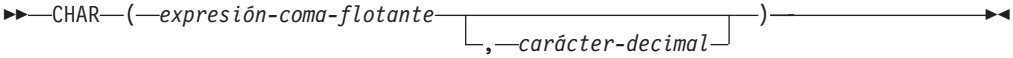


De entero a caracteres:

De decimal a caracteres:



De coma flotante a caracteres:



El esquema es SYSIBM. La signatura SYSFUN.CHAR(*expresión-coma-flotante*) continúa estando disponible. En este caso, el carácter decimal es sensible la configuración local y, por lo tanto, devuelve un punto o una coma, dependiendo del idioma del servidor de la base de datos.

La función CHAR devuelve una representación de serie de caracteres de longitud fija de:

- Una serie de caracteres, si el primer argumento es cualquier tipo de serie de caracteres
- Un valor de fecha y hora, si el primer argumento es una fecha, una hora o una indicación de fecha y hora
- Un número entero, si el primer argumento es SMALLINT, INTEGER o BIGINT
- Un número decimal, si el primer argumento es un número decimal
- Un número de coma flotante de precisión doble, si el primer argumento es DOUBLE o REAL

El primer argumento debe ser de un tipo de datos interno. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

Nota: La expresión CAST también se puede utilizar para devolver una expresión de serie.

El resultado de la función es una serie de caracteres de longitud fija. Si el primer argumento puede ser nulo, el resultado puede ser nulo. Si el primer argumento es nulo, el resultado es el valor nulo.

De caracteres a caracteres

expresión-caracteres
Una expresión que devuelve un valor que es del tipo de datos CHAR, VARCHAR, LONG VARCHAR o CLOB.

entero
El atributo de longitud de la serie de caracteres de longitud fija resultante. El valor debe estar entre 0 y 254.

Si la longitud de la expresión de caracteres es menor que el atributo de longitud del resultado, el resultado se rellena con blancos hasta la longitud del resultado. Si la longitud de la expresión de caracteres es mayor que el atributo de longitud del resultado, el resultado se trunca. Se devuelve un aviso (SQLSTATE 01004) a menos que los caracteres truncados fuesen todos blancos y la expresión de caracteres no fuese una serie larga (LONG VARCHAR o CLOB).

De fecha y hora a caracteres

expresión-fechahora
Una expresión que sea uno de los tres tipos de datos siguientes:

fecha El resultado es la representación de serie de caracteres de la fecha en el formato especificado por el segundo argumento. La longitud del resultado es 10. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

hora El resultado es la representación de serie de caracteres de la hora en el formato especificado por el segundo argumento. La longitud del resultado es de 8. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

indicación de fecha y hora
El resultado es la representación de serie de caracteres de la indicación de fecha y hora. La longitud del resultado es 26. El segundo argumento no es aplicable y no se debe especificar (SQLSTATE 42815).

La página de códigos de la serie es la página de códigos de la base de datos en el servidor de aplicaciones.

De entero a carácter

expresión-entero
Una expresión que devuelve un valor que es de un tipo de datos entero (SMALLINT, INTEGER o BIGINT).

El resultado es la representación de serie de caracteres del argumento en el formato de una constante de enteros de SQL. El resultado consta de *n*

caracteres, que representan los dígitos significativos del argumento, precedido por un signo menos si el argumento es negativo. El resultado está justificado por la izquierda.

- Si el primer argumento es un entero pequeño, la longitud del resultado es de 6.
- Si el primer argumento es un entero grande, la longitud del resultado es de 11.
- Si el primer argumento es un entero superior, la longitud del resultado es de 20.

Si el número de caracteres del resultado es menor que la longitud definida del resultado, el resultado se rellena por la derecha con blancos.

La página de códigos de la serie es la página de códigos de la base de datos en el servidor de aplicaciones.

De decimal a carácter

expresión-decimal
Una expresión que devuelve un valor que es de un tipo de datos decimal. Si se necesita una precisión y escala diferentes, puede utilizarse primero la función escalar DECIMAL para realizar el cambio.

carácter-decimal
Especifica la constante de caracteres de un solo byte que se utiliza para delimitar los dígitos decimales en la serie de caracteres del resultado. La constante de caracteres no puede ser un dígito, el signo más (+), el signo menos (–) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter punto (.).

El resultado es la representación en el formato de una serie de caracteres de longitud fija del argumento. El resultado incluye un carácter decimal y *p* dígitos, donde *p* es la precisión de la *expresión-decimal* precedida por un signo menos si el argumento es negativo. La longitud del resultado es 2 +*p*, donde *p* es la precisión de la *expresión-decimal*. Esto significa que un valor positivo siempre incluirá un blanco de cola.

La página de códigos de la serie es la página de códigos de la base de datos en el servidor de aplicaciones.

De coma flotante a caracteres

expresión-coma-flotante
Una expresión que devuelve un valor que es de un tipo de datos de coma flotante (DOUBLE o REAL).

carácter-decimal
Especifica la constante de caracteres de un solo byte que se utiliza para delimitar los dígitos decimales en la serie de caracteres del resultado. La constante de caracteres no puede ser un dígito, el signo más (+), el signo menos (–) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter punto (.).

El resultado es la representación en el formato de una serie de caracteres de longitud fija del argumento en la forma de una constante de coma flotante. La longitud del resultado es 24. Si el argumento es negativo, el primer carácter del resultado es un signo menos; de lo contrario, el primer carácter es un dígito. Si el valor del argumento es cero, el resultado es 0E0. De lo contrario, el resultado incluye el número más pequeño de caracteres

que puedan representar el valor del argumento de tal modo que la mantisa conste de un solo dígito distinto de cero seguido del *carácter-decimal* y una secuencia de dígitos. Si el número de caracteres del resultado es menor que 24, el resultado se rellena por la derecha con blancos.

La página de códigos de la serie es la página de códigos de la base de datos en el servidor de aplicaciones.

Ejemplos:

- Supongamos que la columna PRSTDATE tiene un valor interno equivalente a 1988-12-25. La función siguiente devuelve el valor '12/25/1988'.

  ```
  CHAR(PRSTDATE, USA)
  ```

- Supongamos que la columna STARTING tiene un valor interno equivalente a 17:12:30 y que la variable del lenguaje principal HOUR_DUR (decimal(6,0)) es una duración en horas con un valor de 050000 (es decir, 5 horas). La función siguiente devuelve el valor '5:12 PM'.

  ```
  CHAR(STARTING, USA)
  ```

  La función siguiente devuelve el valor'10:12 PM'.

  ```
  CHAR(STARTING + :HOUR_DUR, USA)
  ```

- Supongamos que la columna RECEIVED (TIMESTAMP) tiene un valor interno equivalente a la combinación de las columnas PRSTDATE y STARTING. La función siguiente devuelve el valor '1988-12-25-17.12.30.000000'.

  ```
  CHAR(RECEIVED)
  ```

- La columna LASTNAME está definida como VARCHAR(15). La función siguiente devuelve los valores de esta columna como series de caracteres de longitud fija de 10 caracteres de longitud. Los valores de LASTNAME con una longitud superior a los 10 caracteres (excluidos los blancos de cola) se truncan y se devuelve una advertencia.

  ```
  SELECT CHAR(LASTNAME,10) FROM EMPLOYEE
  ```

- La columna EDLEVEL está definida como SMALLINT. La función siguiente devuelve los valores de esta columna como series de caracteres de longitud fija. El valor 18 de EDLEVEL se devuelve como el valor CHAR(6) '18 ' ('18' seguido de cuatro blancos).

  ```
  SELECT CHAR(EDLEVEL) FROM EMPLOYEE
  ```

- La columna SALARY está definida como DECIMAL con una precisión de 9 y una escala de 2. El valor actual (18357.50) debe visualizarse con una coma como el carácter decimal (18357,50). La función siguiente devuelve el valor '00018357,50'.

  ```
  CHAR(SALARY, ',')
  ```

- Los valores de la columna SALARY deben restarse de 20000.25 y visualizarse con el carácter decimal por omisión. La función siguiente devuelve el valor '-0001642.75'.

  ```
  CHAR(20000.25 - SALARY)
  ```

- Supongamos que la variable del lenguaje principal SEASONS_TICKETS está definida como INTEGER y tiene un valor de 10000. La función siguiente devuelve el valor '10000.00'.

  ```
  CHAR(DECIMAL(:SEASONS_TICKETS,7,2))
  ```

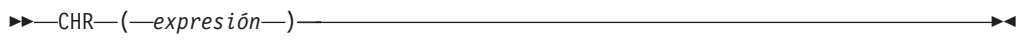
- Supongamos que la variable del lenguaje principal DOUBLE_NUM está definida como DOUBLE y tiene un valor de -987.654321E-35. La función siguiente devuelve el valor '-9.87654321E-33'. Como el tipo de datos del resultado es CHAR(24), hay nueve blancos de cola en el resultado.

```
CHAR(:DOUBLE_NUM)
```

Información relacionada:

- “Expresiones” en la página 181
- “VARCHAR” en la página 456

CHR



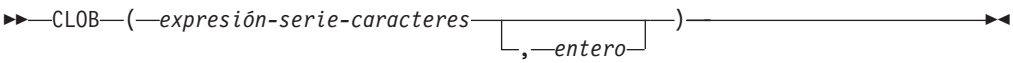
El esquema es SYSFUN.

Devuelve el carácter que tiene el valor del código ASCII especificado por el argumento.

El argumento puede ser INTEGER o SMALLINT. El valor del argumento debe estar entre 0 y 255; de lo contrario, el valor de retorno es nulo.

El resultado de la función es CHAR(1). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

CLOB



El esquema es SYSIBM.

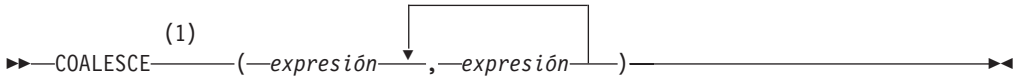
La función CLOB devuelve una representación CLOB de un tipo de serie de caracteres. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

expresión-serie-caracteres
Una *expresión* que devuelve un valor que es una serie de caracteres.

entero
Un valor entero que especifica el atributo de longitud del tipo de datos CLOB resultante. El valor debe estar entre 0 y 2 147 483 647. Si no se especifica *entero*, la longitud del resultado es la misma que la longitud del primer argumento.

El resultado de la función es CLOB. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

COALESCE



Notas:

1 VALUE es sinónimo de COALESCE.

El esquema es SYSIBM.

COALESCE devuelve el primer argumento que no es nulo.

Los argumentos se evalúan en el orden en que se especifican, y el resultado de la función es el primer argumento que no es nulo. El resultado sólo puede ser nulo si todos los argumentos pueden ser nulos, y el resultado sólo es nulo si todos los argumentos son nulos. El argumento seleccionado se convierte, si es necesario, a los atributos del resultado.

Los argumentos deben ser compatibles. Puede ser de un tipo de datos interno o definido por el usuario. (Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como esta función acepta cualquier tipo de datos compatible como argumento, no es necesario crear signaturas adicionales para dar soporte a los tipos de datos diferenciados definidos por el usuario).

Ejemplos:

- Cuando se seleccionan todos los valores de todas las filas de la tabla DEPARTMENT, si falta el director del departamento (MGRNO) (es decir, es nulo), se ha de devolver un valor de 'ABSENT'.

```
SELECT DEPTNO, DEPTNAME, COALESCE(MGRNO, 'ABSENT'), ADMRDEPT
 FROM DEPARTMENT
```

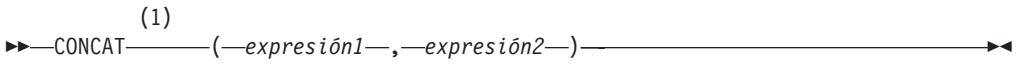
- Cuando se selecciona el número de empleado (EMPNO) y el salario (SALARY) de todas las filas de la tabla EMPLOYEE, si falta el salario (es decir, es nulo), se ha de devolver un valor de cero.

```
SELECT EMPNO, COALESCE(SALARY, 0)
  FROM EMPLOYEE
```

Información relacionada:

- "Reglas para los tipos de datos del resultado" en la página 125

CONCAT



Notas:

1 || se utiliza como sinónimo de CONCAT.

El esquema es SYSIBM.

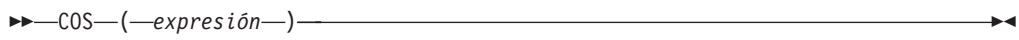
Devuelve la concatenación de dos argumentos de serie. Los dos argumentos deben ser de tipos compatibles.

El resultado de la función es una serie. La longitud es la suma de las longitudes de los dos argumentos. Si cualquier argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Información relacionada:

• "Expresiones" en la página 181

COS



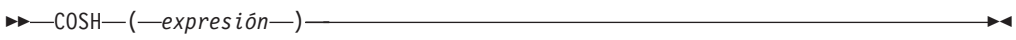
El esquema es SYSIBM. (La versión SYSFUN de la función COS continúa estando disponible).

Devuelve el coseno del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo numérico interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

COSH



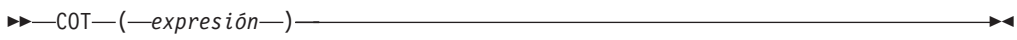
El esquema es SYSIBM.

Devuelve el coseno hiperbólico del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

COT



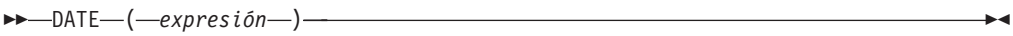
El esquema es SYSIBM. (La versión SYSFUN de la función COT continúa estando disponible).

Devuelve la cotangente del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo numérico interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

DATE



El esquema es SYSIBM.

La función DATE devuelve una fecha de un valor.

El argumento debe ser una fecha, indicación de fecha y hora, número positivo menor o igual que 3 652 059, representación de serie válida de una fecha o una indicación de fecha y hora o una serie de longitud 7 que no sea CLOB, LONG VARCHAR ni LONG VARGRAPHIC.

Sólo las bases de datos Unicode dan soporte a un argumento que es una representación de serie gráfica de una fecha o una indicación de fecha y hora. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

Si el argumento es una serie de longitud 7, debe representar una fecha válida en el formato *aaaannn*, donde *aaaa* son los dígitos que indican el año y *nnn* los son dígitos entre 001 y 366, que indican un día de dicho año.

El resultado de la función es una fecha. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una fecha, una indicación de fecha y hora o una representación válida en el formato de una serie de una fecha o indicación de fecha y hora:
  - El resultado es la parte correspondiente a la fecha del valor.
- Si el argumento es un número:
  - El resultado es la fecha de *n*-1 días después de 1 de enero de 0001, donde *n* es la parte integral del número.
- Si el argumento es una serie con una longitud de 7:
  - El resultado es la fecha representada por la serie.

Ejemplos:

Supongamos que la columna RECEIVED (indicación de fecha y hora) tiene un valor interno equivalente a '1988-12-25-17.12.30.000000'.

- Este ejemplo da como resultado una representación interna de '1988-12-25'.
  ```
  DATE(RECEIVED)
  ```
- Este ejemplo da como resultado una representación interna de '1988-12-25'.
  ```
  DATE('1988-12-25')
  ```
- Este ejemplo da como resultado una representación interna de '1988-12-25'.
  ```
  DATE('25.12.1988')
  ```
- Este ejemplo da como resultado una representación interna de '0001-02-04'.
  ```
  DATE(35)
  ```

DAY

El esquema es SYSIBM.

La función DAY devuelve la parte correspondiente al día de un valor.

El argumento debe ser una fecha, una indicación de fecha y hora, una duración de fecha, una duración de una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una fecha, una indicación de fecha y hora o una representación válida de una fecha o indicación de fecha y hora en forma de serie:
  - El resultado es la parte correspondiente al día del valor, que es un entero entre 1 y 31.
- Si el argumento es una duración de fecha o duración de la indicación de fecha y hora:
  - El resultado es la parte correspondiente al día del valor, que es un entero entre –99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplos:

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal END_DAY (smallint) en el día en que está planificado que el proyecto WELD LINE PLANNING (PROJNAME) finalice (PRENDATE).

  ```
  SELECT DAY(PRENDATE)
    INTO :END_DAY
    FROM PROJECT
    WHERE PROJNAME = 'WELD LINE PLANNING'
  ```

  Da como resultado que END_DAY se establece en 15 cuando se utiliza la tabla de ejemplo.
- Supongamos que la columna DATE1 (fecha) tiene un valor interno equivalente a 2000-03-15 y la columna DATE2 (fecha) tiene un valor interno equivalente a 1999-12-31.

  ```
  DAY(DATE1 - DATE2)
  ```

  Da como resultado el valor 15.

DAYNAME

El esquema es SYSFUN.

Devuelve una serie que combina caracteres en mayúsculas y minúsculas que contiene el nombre del día (por ejemplo, viernes) correspondiente a la parte del día del argumento, según el entorno local en el momento en que se inició la base de datos.

El argumento debe ser una fecha, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o de una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es VARCHAR(100). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

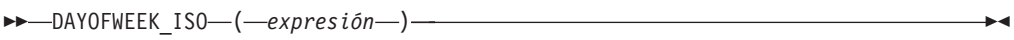
DAYOFWEEK

Devuelve el día de la semana del argumento como un valor entero en el rango de 1 a 7, donde 1 representa el Domingo.

El argumento debe ser una fecha, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o de una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DAYOFWEEK_ISO



El esquema es SYSFUN.

Devuelve el día de la semana del argumento, en forma de valor entero comprendido dentro del rango 1-7, donde 1 representa el lunes.

El argumento debe ser una fecha, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o de una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DAYOFYEAR

El esquema es SYSFUN.

Devuelve el día del año del argumento como un valor entero en el rango de 1 a 366.

El argumento debe ser una fecha, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o de una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DAYS

El esquema es SYSIBM.

La función DAYS devuelve una representación de entero de una fecha.

El argumento debe ser una fecha, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o de una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

El resultado es 1 más que el número de días desde el 1 de enero de 0001 hasta *D*, donde *D* es la fecha que podría darse si se aplicase la función DATE al argumento.

Ejemplos:

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal EDUCATION_DAYS (int) en el número de días transcurridos (PRENDATE - PRSTDATE) estimados para el proyecto (PROJNO) 'IF2000'.

  ```
  SELECT DAYS(PRENDATE) - DAYS(PRSTDATE)
    INTO :EDUCATION_DAYS
    FROM PROJECT
    WHERE PROJNO = 'IF2000'
  ```

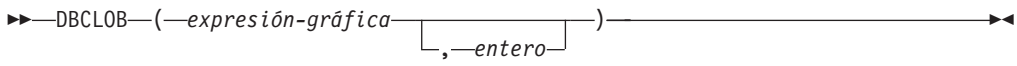
  El resultado de EDUCATION_DAYS se define en 396.

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal TOTAL_DAYS (int) en la suma de los días transcurridos (PRENDATE - PRSTDATE) estimados para todos los proyectos del departamento (DEPTNO) 'E21'.

  ```
  SELECT SUM(DAYS(PRENDATE) - DAYS(PRSTDATE))
    INTO :TOTAL_DAYS
    FROM PROJECT
    WHERE DEPTNO = 'E21'
  ```

  Da como resultado que TOTAL_DAYS se establece en 1584 cuando se utiliza la tabla de ejemplo.

DBCLOB



El esquema es SYSIBM.

La función DBCLOB devuelve una representación DBCLOB de un tipo de serie gráfica.

En una base de datos Unicode, si un argumento proporcionado es una serie de caracteres, se convertirá a una serie gráfica antes de que se ejecute la función. Cuando la serie de salida se trunca, de forma que el último carácter es un carácter de sustitución elevado, dicho carácter:

- Se deja tal cual, si el argumento proporcionado es una serie de caracteres
- O se convierte al carácter en blanco (X'0020'), si el argumento proporcionado es una serie gráfica

No confíe en estos comportamientos, porque podrían cambiar en los releases futuros.

El resultado de la función es DBCLOB. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

expresión-gráfica
Una expresión que devuelve un valor que es una serie gráfica.

entero
Un valor entero que especifica el atributo de longitud del tipo de datos DBCLOB resultante. El valor debe estar comprendido entre 0 y 1 073 741 823. Si no se especifica *entero*, la longitud del resultado es la misma que la longitud del primer argumento.

DBPARTITIONNUM

El esquema es SYSIBM.

La función DBPARTITIONNUM devuelve el número de partición de la fila. Por ejemplo, si se utiliza en una cláusula SELECT, devuelve el número de partición para cada fila de la tabla que se ha utilizado para formar el resultado de la sentencia SELECT.

El número de partición devuelto en las tablas y variables de transición procede de los valores de transición actuales de las columnas de claves de particionamiento. Por ejemplo, en un activador BEFORE INSERT, la función devolverá el número de partición proyectado teniendo en cuenta los valores actuales de las nuevas variables de transición. No obstante, un activador BEFORE INSERT subsiguiente puede modificar los valores de las columnas de claves de particionamiento. De este modo, el número de partición final de la fila cuando se inserte en la base de datos puede ser diferente del valor proyectado.

El argumento debe ser el nombre calificado o no calificado de una columna de una tabla. La columna puede tener cualquier tipo de datos. (Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos como argumento, no es necesario crear signaturas adicionales para dar soporte a los tipos diferenciados definidos por el usuario). Si *nombre-columna* hace referencia a una columna de una vista, la expresión de la vista para la columna debe hacer referencia a una columna de la tabla base principal y la vista debe ser suprimible. Una expresión de tabla anidada o común sigue las mismas reglas que una vista.

La fila específica (y tabla) para la que se devuelve el número de partición mediante la función DBPARTITIONNUM se determina por el contexto de la sentencia de SQL que utiliza la función.

El tipo de datos del resultado es INTEGER y nunca es nulo. Puesto que se devuelve información a nivel de fila, los resultados son los mismos, sin tener en cuenta las columnas que se especifican para la tabla. Si no hay ningún archivo `db2nodes.cfg`, el resultado es 0.

La función DBPARTITIONNUM no puede utilizarse en tablas duplicadas, dentro de restricciones de comprobación ni en la definición de columnas generadas (SQLSTATE 42881).

Para compatibilidad con versiones anteriores a la Versión 8, la palabra clave NODENUMBER puede sustituirse por DBPARTITIONNUM.

Ejemplos:

- Cuente el número de filas en las que la fila para un EMPLOYEE está en una partición diferente de la descripción del departamento del empleado en DEPARTMENT.

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E
  WHERE D.DEPTNO=E.WORKDEPT
  AND DBPARTITIONNUM(E.LASTNAME) <> DBPARTITIONNUM(D.DEPTNO)
```

- Una las tablas EMPLOYEE y DEPARTMENT donde las filas de las dos tablas estén en la misma partición.

```
SELECT * FROM DEPARTMENT D, EMPLOYEE E
  WHERE DBPARTITIONNUM(E.LASTNAME) = DBPARTITIONNUM(D.DEPTNO)
```

- Anote cronológicamente el número de empleado y el número de partición proyectado de la nueva fila en una tabla denominada EMPINSERTLOG1 para cualquier inserción de empleados creando un activador BEFORE en la tabla EMPLOYEE.

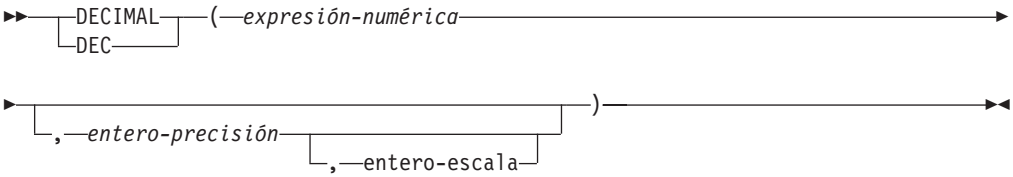
```
 CREATE TRIGGER EMPINSLOGTRIG1
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AW NEWTABLE
 FOR EACH ROW
 INSERT INTO EMPINSERTLOG1
 VALUES(NEWTABLE.EMPNO, DBPARTITIONNUM
 (NEWTABLE.EMPNO))
```

Información relacionada:

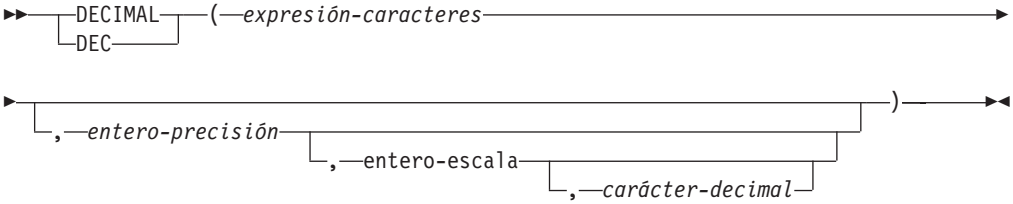
- "Sentencia CREATE VIEW" en la publicación *Consulta de SQL, Volumen 2*

DECIMAL

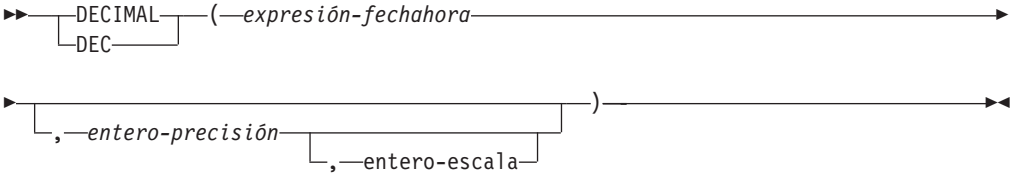
De numérico a decimal :



De carácter a decimal:



De fecha y hora a decimal:



El esquema es SYSIBM.

La función DECIMAL devuelve una representación decimal de:

- Un número
- Una representación de serie de caracteres de un número decimal
- Una representación de serie de caracteres de un número entero
- Una representación de serie de caracteres de un número de coma flotante
- Un valor de fecha y hora, si el argumento es una fecha, una hora o una indicación de fecha y hora

En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un número decimal con precisión *p* y escala *s*, donde *p* y *s* son el segundo y el tercer argumento, respectivamente. Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

De numérico a decimal

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico.

entero-precisión
Una constante de enteros con un valor en el rango de 1 a 31.

El valor por omisión para *entero-precisión* depende del tipo de datos de *expresión-numérica*:

- 15 para coma flotante y decimal
- 19 para entero superior
- 11 para entero grande
- 5 para entero pequeño.

entero-escala
Una constante de enteros en el rango de 0 al valor de *entero-precisión*. El valor por omisión es cero.

El resultado es el mismo número que sería si se asignase el primer argumento a una columna o variable decimal con precisión *p* y escala *s*, donde *p* y *s* son el segundo y el tercer argumento, respectivamente. Se produce un error si el número de dígitos decimales significativos necesarios para representar la parte correspondiente a los enteros es mayor que *p*−*s*.

De carácter a decimal

expresión-caracteres
Una *expresión* que devuelve un valor que es una serie de caracteres con una longitud no mayor que la longitud máxima de una constante de caracteres (4.000 bytes). No puede tener un tipo de datos CLOB ni LONG VARCHAR. Los blancos iniciales o de cola se eliminan de la serie. La subserie resultante debe ajustarse a las reglas para formar una constante decimal o de entero SQL (SQLSTATE 22018).

La *expresión-caracteres* se convierte a la página de códigos de la base de datos si es necesario que coincida con la página de códigos de la constante *carácter-decimal*.

entero-precisión
Una constante de enteros con un valor en el rango de 1 a 31 que especifica la precisión del resultado. Si no se especifica, el valor por omisión es 15.

entero-escala
Una constante de enteros con un valor en el rango entre 0 y *entero-precisión* que especifica la escala del resultado. Si no se especifica, el valor por omisión es 0.

carácter-decimal
Especifica la constante de caracteres de un solo byte utilizada para delimitar los dígitos decimales en *expresión-caracteres* de la parte correspondiente a los enteros del número. El carácter no puede ser un dígito, el signo más (+), el signo menos (–) ni un blanco y puede aparecer como máximo una vez en *expresión-caracteres* (SQLSTATE 42815).

El resultado es un número decimal con la precisión *p* y la escala *s*, donde *p* y *s* son el segundo y el tercer argumento, respectivamente. Los dígitos se truncan por el final del número decimal si el número de dígitos a la derecha del carácter decimal es mayor que la escala. Se produce un error si el número de dígitos significativos a la izquierda del carácter decimal (la parte correspondiente a los enteros del número) de *expresión-caracteres* es

mayor que *p–s* (SQLSTATE 22003).El carácter decimal por omisión no es válido en la subserie si se especifica un valor diferente para el argumento *carácter-decimal* (SQLSTATE 22018).

De fecha y hora a decimal

expresión-fechahora
Una expresión que sea uno de los tipos de datos siguientes:

- DATE. El resultado es un valor DECIMAL(8,0) que representa la fecha como *aaaammdd*.
- TIME. El resultado es un valor DECIMAL(6,0) que representa la hora como *hhmmss*.
- TIMESTAMP. El resultado es un valor DECIMAL(20,6) que representa la indicación de fecha y hora como *aaaammddhhmmss*.

Este función permite que el usuario especifique una precisión o una precisión y una escala. Sin embargo, una escala no puede especificarse sin especificar una precisión. El valor por omisión para (precisión,escala) es (8,0) para DATE, (6,0) para TIME y (20,6) para TIMESTAMP.

El resultado es un número decimal con la precisión *p* y la escala *s*, donde *p* y *s* son el segundo y el tercer argumento, respectivamente. Los dígitos se truncan por el final si el número de dígitos a la derecha del carácter decimal es mayor que la escala. Se produce un error si el número de dígitos significativos a la izquierda del carácter decimal (la parte correspondiente a los enteros del número) de *expresión-fechahora* es mayor que *p–s* (SQLSTATE 22003).

Ejemplos:

- Utilice la función DECIMAL para forzar a que se devuelva un tipo de datos DECIMAL (con una precisión de 5 y una escala de 2) en una lista-selección para la columna EDLEVEL (tipo de datos = SMALLINT) en la tabla EMPLOYEE. La columna EMPNO debe aparecer también en la lista de selección.

```
SELECT EMPNO, DECIMAL(EDLEVEL,5,2)
  FROM EMPLOYEE
```

- Supongamos que la variable del lenguaje principal PERIOD es de tipo INTEGER. En este caso, para utilizar su valor como duración de fecha debe convertirse a decimal(8,0).

```
SELECT PRSTDATE + DECIMAL(:PERIOD,8)
  FROM PROJECT
```

- Supongamos que las actualizaciones en la columna SALARY se entran mediante una ventana como una serie de caracteres que utiliza la coma como carácter decimal (por ejemplo, el usuario entra 21400,50). Cuando se ha validado por la aplicación, se asigna a la variable del lenguaje principal newsalary definida como CHAR(10).

```
UPDATE STAFF
  SET SALARY = DECIMAL(:newsalary, 9, 2, ',')
  WHERE ID = :empid;
```

  El valor de newsalary se convierte en 21400.50.

- Añada el carácter decimal por omisión (.) al valor.

```
DECIMAL('21400,50', 9, 2, '.')
```

  Falla porque se especifica un punto (.) como el carácter decimal, pero aparece una coma (,) en el primer argumento como delimitador.

- Supongamos que la columna STARTING (hora) tiene un valor interno equivalente a '12:10:00'.

  ```
  DECIMAL(STARTING)
  ```

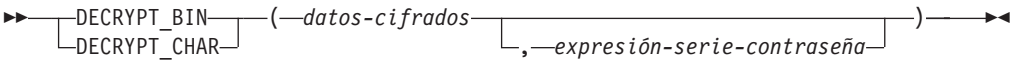
  da como resultado el valor `121 000`.
- Supongamos que la columna RECEIVED (indicación de fecha y hora) tiene un valor interno equivalente a '1988-12-22-14.07.21.136421'.

  ```
  DECIMAL(RECEIVED)
  ```

  da como resultado el valor `19 881 222 140 721.136421`.
- La tabla siguiente muestra el resultado decimal y la precisión y la escala resultante para varios valores de entrada de fecha y hora.

DECIMAL(argumentos)	Precisión y escala	Resultado
DECIMAL(2000-03-21)	(8,0)	20000321
DECIMAL(2000-03-21, 10)	(10,0)	20000321
DECIMAL(2000-03-21, 12, 2)	(12,2)	20000321.00
DECIMAL(12:02:21)	(6,0)	120221
DECIMAL(12:02:21, 10)	(10,0)	120221
DECIMAL(12:02:21, 10, 2)	(10,2)	120221.00
DECIMAL(2000-03-21-12.02.21.123456)	(20, 6)	20000321120221.123456
DECIMAL(2000-03-21-12.02.21.123456, 23)	(23, 6)	20000321120221.123456
DECIMAL(2000-03-21-12.02.21.123456, 23, 4)	(23, 4)	20000321120221.1234

DECRYPT_BIN y DECRYPT_CHAR



El esquema es SYSIBM.

Las funciones DECRYPT_BIN y DECRYPT_CHAR devuelven un valor que es el resultado del descifrado de *datos-cifrados*. La contraseña utilizada para el descifrado es el valor de *expresión-serie-contraseña* o el valor de ENCRYPTION PASSWORD asignado por la sentencia SET ENCRYPTION PASSWORD. Las funciones DECRYPT_BIN y DECRYPT_CHAR sólo pueden descifrar valores que se han cifrado utilizando la función ENCRYPT (SQLSTATE 428FE).

datos-cifrados
Una expresión que devuelve un valor CHAR FOR BIT DATA o VARCHAR FOR BIT DATA como una serie de datos cifrada completa. La serie de datos se tiene que haber cifrado utilizando la función ENCRYPT.

expresión-serie-contraseña
Una expresión que devuelve un valor CHAR o VARCHAR con un mínimo de 6 bytes y no más de 127 bytes (SQLSTATE 428FC). Esta expresión debe ser la misma contraseña utilizada para cifrar los datos o, de lo contrario, el descifrado producirá un error (SQLSTATE 428FD). Si el valor del argumento de contraseña es nulo o no se proporciona, los datos se cifrarán utilizando el valor de ENCRYPTION PASSWORD, que tiene que haberse establecido para la sesión (SQLSTATE 51039).

El resultado de la función DECRYPT_BIN es VARCHAR FOR BIT DATA. El resultado de la función DECRYPT_CHAR es VARCHAR. Si los *datos-cifrados* incluían una indicación, la función no devuelve la indicación. El atributo de longitud del resultado es la longitud del tipo de datos de los *datos-cifrados* menos 8 bytes. La longitud real del valor devuelto por la función coincidirá con la longitud de la serie original que se ha cifrado. Si *datos-cifrados* incluye bytes más allá de la serie cifrada, la función no devolverá estos bytes.

Si el primer argumento puede ser nulo, el resultado puede ser nulo. Si el primer argumento es nulo, el resultado es el valor nulo.

Si los datos se descifran en un sistema diferente que utiliza una página de códigos diferente de la página de códigos en la que se han cifrado los datos, puede que se produzca una expansión al convertir el valor descifrado a la página de códigos de la base de datos. En dichas situaciones, el valor *datos-cifrados* debe calcularse en una serie VARCHAR con un número mayor de bytes.

Ejemplos:

Ejemplo 1: Este ejemplo utiliza el valor de ENCRYPTION PASSWORD para retener la contraseña de cifrado.

```
SET ENCRYPTION PASSWORD = 'Ben123';
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832');
SELECT DECRYPT_CHAR(SSN)
  FROM EMP;
```

Esto devuelve el valor '289-46-8832'.

Ejemplo 2: Este ejemplo pasa explícitamente la contraseña de cifrado.

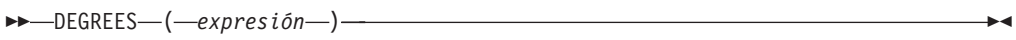
```
INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832','Ben123','');
SELECT DECRYPT(SSN,'Ben123')
  FROM EMP;
```

Este ejemplo devuelve el valor '289-46-8832'.

Información relacionada:

- "Sentencia SET ENCRYPTION PASSWORD" en la publicación *Consulta de SQL, Volumen 2*
- "ENCRYPT" en la página 356
- "GETHINT" en la página 363

DEGREES



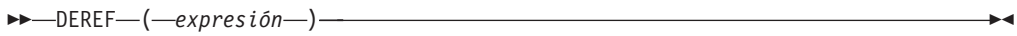
El esquema es SYSFUN.

Devuelve el número de grados convertidos del argumento expresado en radianes.

El argumento puede ser de cualquier tipo numérico interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DEREF



La función DEREF devuelve una instancia del tipo de destino del argumento.

El argumento puede ser cualquier valor con un tipo de datos de referencia que tenga un ámbito definido (SQLSTATE 428DT).

El tipo de datos estático del resultado es el tipo de destino del argumento. El tipo de datos dinámico del resultado es un subtipo del tipo de destino del argumento. El resultado puede ser nulo. El resultado es un valor nulo si *expresión* es un valor nulo o si *expresión* es una referencia que no tiene un OID correspondiente en la tabla de destino.

El resultado es una instancia del subtipo del tipo de destino de la referencia. El resultado se determina buscando la fila de la tabla de destino o vista de destino de la referencia que tenga un identificador de objeto que se corresponda con el valor de la referencia. El tipo de esta fila determina el tipo dinámico del resultado. Puesto que el tipo del resultado puede estar basado en una fila de una subtabla o subvista de la tabla de destino o vista de destino, el ID de autorización de la sentencia debe tener un privilegio SELECT sobre la tabla de destino y todas sus subtablas o sobre la vista de destino y todas sus subvistas (SQLSTATE 42501).

Ejemplos:

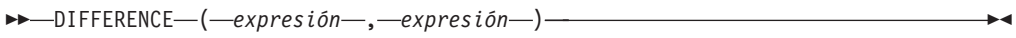
Supongamos que EMPLOYEE es una tabla de tipo EMP, y que su columna de identificador de objeto se llama EMPID. En este caso, la consulta siguiente devuelve un objeto de tipo EMP (o uno de sus subtipos) para cada fila de la tabla EMPLOYEE (y de sus subtablas). Para ejecutar esta consulta es necesario tener privilegio SELECT sobre EMPLOYEE y todas sus subtablas.

```
SELECT DEREF(EMPID) FROM EMPLOYEE
```

Información relacionada:

- "TYPE_NAME" en la página 452

DIFFERENCE



El esquema es SYSFUN.

Devuelve un valor de 0 a 4 que representa la diferencia entre los sonidos de dos series basándose en la aplicación de la función SOUNDEX en las series. El valor 4 es la mejor coincidencia de sonido posible.

Los argumentos pueden ser series de caracteres que sean CHAR o VARCHAR de un máximo de 4.000 bytes. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

```
VALUES (DIFFERENCE('CONSTRAINT','CONSTANT'),SOUNDEX('CONSTRAINT'),
SOUNDEX('CONSTANT')),
(DIFFERENCE('CONSTRAINT','CONTRITE'),SOUNDEX('CONSTRAINT'),
SOUNDEX('CONTRITE'))
```

Este ejemplo devuelve lo siguiente.

```
  1           2    3
----------- ---- ----
          4 C523 C523
          2 C523 C536
```

En la primera fila, las palabras tienen el mismo resultado de SOUNDEX, mientras que en la segunda fila las palabras sólo tienen algún parecido.

Información relacionada:

• "SOUNDEX" en la página 428

DIGITS

El esquema es SYSIBM.

La función DIGITS devuelve una representación de serie de caracteres de un número.

El argumento debe ser una expresión que devuelva un valor con el tipo SMALLINT, INTEGER, BIGINT o DECIMAL.

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

El resultado de la función es una serie de caracteres de longitud fija que representa el valor absoluto del argumento sin tener en cuenta su escala. El resultado no incluye el signo ni el carácter decimal. En su lugar, consta exclusivamente de dígitos, incluyendo, si es necesario, ceros iniciales para rellenar la serie. La longitud de la serie es:

- 5 si el argumento es un entero pequeño
- 10 si el argumento es un entero grande
- 19 si el argumento es un entero superior
- *p* si el argumento es un número decimal con una precisión de *p*.

Ejemplos:

- Supongamos que una tabla llamada TABLEX contiene una columna INTEGER llamada INTCOL que contiene números de 10 dígitos. Liste las cuatro combinaciones de dígitos de los cuatro primeros dígitos de la columna INTCOL.

  ```
  SELECT DISTINCT SUBSTR(DIGITS(INTCOL),1,4)
    FROM TABLEX
  ```

- Supongamos que la columna COLUMNX tiene el tipo de datos DECIMAL(6,2) y que uno de sus valores es -6.28. Entonces, para este valor:

  ```
  DIGITS(COLUMNX)
  ```

  devuelve el valor '000628'.

  El resultado es una serie de longitud seis (la precisión de la columna) con ceros iniciales que rellenan la serie hasta esta longitud. No aparecen ni el signo ni la coma decimal en el resultado.

DLCOMMENT

El esquema es SYSIBM.

La función DLCOMMENT devuelve el valor del comentario, si existe, de un valor DATALINK.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos de DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Prepare una sentencia para seleccionar la fecha, la descripción y el comentario (en el enlace de la columna ARTICLES) de la tabla HOCKEY_GOALS. Las filas a seleccionar son las correspondientes a los goles marcados por cualquiera de los dos hermanos Richard (Maurice o Henri).

```
stmtvar = "SELECT DATE_OF_GOAL, DESCRIPTION, DLCOMMENT(ARTICLES)
             FROM HOCKEY_GOALS
             WHERE BY_PLAYER = 'Maurice Richard'
             OR BY_PLAYER = 'Henri Richard' ";
EXEC SQL PREPARE HOCKEY_STMT FROM :stmtvar;
```

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

```
DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','Un comentario')
```

  la siguiente función que realiza una operación con este valor:

```
DLCOMMENT(COLA)
```

  devolverá el valor:

```
Un comentario
```

DLLINKTYPE

El esquema es SYSIBM.

La función DLLINKTYPE devuelve el valor de tipoenlace de un valor DATALINK.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(4). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
  ```

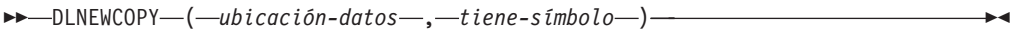
  la siguiente función que realiza una operación con este valor:

  ```
  DLLINKTYPE(COLA)
  ```

  devolverá el valor:

  ```
  URL
  ```

DLNEWCOPY



El esquema es SYSIBM.

La función DLNEWCOPY devuelve un valor DATALINK que tiene un atributo que indica que se ha modificado el archivo referenciado. Si se asigna este valor a una columna DATALINK como resultado de una sentencia SQL UPDATE, se comunica a DB2 que ha finalizado una actualización del archivo enlazado. Si la columna DATALINK está definida con RECOVERY YES, la nueva versión del archivo enlazado se archiva de forma asíncrona. Si se asigna este valor a una columna DATALINK como resultado de una sentencia SQL INSERT, se devuelve un error (SQLSTATE 428D1).

ubicación-datos
Una expresión VARCHAR(200) que especifica un serie de caracteres de longitud variable que contiene un valor de URL completo. El valor puede haberse obtenido anteriormente mediante una sentencia SELECT utilizando la función DLURLCOMPLETEWRITE.

tiene-símbolo
Un valor INTEGER que indica si la ubicación de los datos contiene un símbolo de escritura.

0 La ubicación de los datos no contiene un símbolo de escritura.

1 La ubicación de los datos contiene un símbolo de escritura.

Se produce un error si el valor no es ni 0 ni 1 (SQLSTATE 42815), o si el símbolo intercalado en la ubicación de los datos no es válido (SQLSTATE 428D1).

El resultado de la función es un valor DATALINK sin el símbolo de escritura. Ni *ubicación-datos* ni *tiene-símbolo* puede ser nulo.

Para una columna DATALINK definida con WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE, el símbolo de escritura debe estar en la ubicación de los datos para que se complete la sentencia SQL UPDATE (SQLSTATE 428D1). Por otra parte, para WRITE PERMISSION ADMIN NOT REQUIRING TOKEN FOR UPDATE no se requiere el símbolo de escritura pero se permite en la ubicación de los datos.

Para una columna DATALINK definida con WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE, el símbolo de escritura debe ser el mismo que el utilizado para abrir el archivo especificado, si se había abierto (SQLSTATE 428D1).

Para cualquier columna WRITE PERMISSION ADMIN, aunque el símbolo de escritura haya caducado, el símbolo sigue considerándose válido siempre que se utilice el mismo símbolo para abrir el archivo especificado para acceso de escritura.

En caso de que no se haya realizado ninguna actualización del archivo o el archivo DATALINK esté enlazado con otras opciones como, por ejemplo, WRITE PERMISSION BLOCKED/FS o NO LINK CONTROL, esta función se comportará como DLVALUE.

Ejemplos:

- Dado un valor DATALINK que se haya insertado en la columna COLA (definida con WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE) en la tabla TBLA utilizando la función escalar:

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
  ```

  Utilice la función escalar DLURLCOMPLETEWRITE para buscar el valor:

  ```
  SELECT DLURLCOMPLETEWRITE(COLA)
    FROM TBLA
    WHERE ...
  ```

  Devuelve:

  ```
  HTTP://DLFS.ALMADEN.IBM.COM/x/y/****************;a.b
  ```

  donde ********** representa el símbolo de escritura.

  Utilice el valor anterior para localizar y actualizar el contenido del archivo. Emita la siguiente sentencia SQL UPDATE para indicar que el archivo se ha modificado de forma satisfactoria:

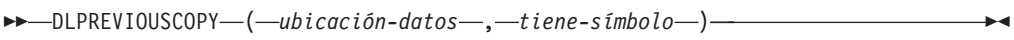
  ```
  UPDATE TBLA
    SET COLA = DLNEWCOPY('http://dlfs.almaden.ibm.com/x/y/********
      ********;a.b', 1)
    WHERE ...
  ```

  donde ********** representa el mismo símbolo de escritura utilizado para modificar el archivo al que el valor de URL hace referencia. Observe que si COLA está definido con WRITE PERMISSION ADMIN NOT REQUIRING TOKEN FOR UPDATE, el símbolo de escritura no es necesario en el ejemplo anterior.

- El valor del segundo argumento (*tiene-símbolo*) puede sustituirse por la sentencia CASE siguiente. Supongamos que el valor de URL está contenido en una variable denominada *url_file*. Emita la siguiente sentencia SQL UPDATE para indicar que el archivo se ha modificado de forma satisfactoria:

  ```
  EXEC SQL UPDATE TBLA
    SET COLA = DLNEWCOPY(:url_file,
      (CASE
        WHEN LENGTH(:url_file) = LENGTH(DLURLCOMPLETEONLY(COLA))
        THEN 0
        ELSE 1
       END))
    WHERE ...
  ```

DLPREVIOUSCOPY



El esquema es SYSIBM.

La función DLPREVIOUSCOPY devuelve un valor DATALINK que tiene un atributo que indica que debería restaurarse la versión anterior del archivo. Si se asigna este valor a una columna DATALINK como resultado de una sentencia SQL UPDATE, provoca que DB2 restaure el archivo enlazado de la versión confirmada anteriormente. Si se asigna este valor a una columna DATALINK como resultado de una sentencia SQL INSERT, se devuelve un error (SQLSTATE 428D1).

ubicación-datos
Una expresión VARCHAR(200) que especifica un serie de caracteres de longitud variable que contiene un valor de URL completo. El valor puede haberse obtenido anteriormente mediante una sentencia SELECT utilizando la función DLURLCOMPLETEWRITE.

tiene-símbolo
Un valor INTEGER que indica si la ubicación de los datos contiene un símbolo de escritura.

0 La ubicación de los datos no contiene un símbolo de escritura.

1 La ubicación de los datos contiene un símbolo de escritura.

Se produce un error si el valor no es ni 0 ni 1 (SQLSTATE 42815), o si el símbolo intercalado en la ubicación de los datos no es válido (SQLSTATE 428D1).

El resultado de la función es un valor DATALINK sin el símbolo de escritura. Ni *ubicación-datos* ni *tiene-símbolo* puede ser nulo.

Para una columna DATALINK definida con WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE, el símbolo de escritura debe estar en la ubicación de los datos para que se complete la sentencia SQL UPDATE (SQLSTATE 428D1). Por otra parte, para WRITE PERMISSION ADMIN NOT REQUIRING TOKEN FOR UPDATE no se requiere el símbolo de escritura pero se permite en la ubicación de los datos.

Para una columna DATALINK definida con WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE, el símbolo de escritura debe ser el mismo que el utilizado para abrir el archivo especificado, si se había abierto (SQLSTATE 428D1).

Para cualquier columna WRITE PERMISSION ADMIN, aunque el símbolo de escritura haya caducado, el símbolo sigue considerándose válido siempre que se utilice el mismo símbolo para abrir el archivo especificado para acceso de escritura.

Ejemplos:

- Dado un valor DATALINK que se haya insertado en la columna COLA (definida con WRITE PERMISSION ADMIN REQUIRING TOKEN FOR UPDATE y RECOVERY YES) en la tabla TBLA utilizando la función escalar:

```
DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
```

Utilice la función escalar DLURLCOMPLETEWRITE para buscar el valor:

```
SELECT DLURLCOMPLETEWRITE(COLA)
  FROM TBLA
  WHERE ...
```

Devuelve:

```
HTTP://DLFS.ALMADEN.IBM.COM/x/y/****************;a.b
```

donde ********** representa el símbolo de escritura.

Utilice el valor anterior para localizar y actualizar el contenido del archivo. Emita la sentencia SQL UPDATE siguiente para restituir los cambios en el archivo y restaurarlo en la versión confirmada anterior.

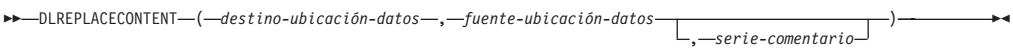
```
UPDATE TBLA
  SET COLA = DLPREVIOUSCOPY('http://dlfs.almaden.ibm.com/x/y/********
    ********;a.b', 1)
  WHERE ...
```

donde ********** representa el mismo símbolo de escritura utilizado para modificar el archivo al que el valor de URL hace referencia. Observe que si COLA está definido con WRITE PERMISSION ADMIN NOT REQUIRING TOKEN FOR UPDATE, el símbolo de escritura no es necesario en el ejemplo anterior.

- El valor del segundo argumento (*tiene-símbolo*) puede sustituirse por la sentencia CASE siguiente. Supongamos que el valor de URL está contenido en una variable denominada *url_file*. Emita la sentencia SQL UPDATE siguiente para restituir los cambios en el archivo y restaurarlo en la versión confirmada anterior.

```
EXEC SQL UPDATE TBLA
  SET COLA = DLPREVIOUSCOPY(:url_file,
    (CASE
      WHEN LENGTH(:url_file) = LENGTH(DLURLCOMPLETEONLY(COLA))
      THEN 0
      ELSE 1
     END))
  WHERE ...
```

DLREPLACECONTENT



El esquema es SYSIBM.

La función DLREPLACECONTENT devuelve un valor DATALINK. Cuando la función está al lado derecho de una cláusula SET de una sentencia UPDATE o está en una cláusula VALUES dentro de una sentencia INSERT, la asignación del valor devuelto provoca la sustitución del contenido de un archivo por otro archivo y la creación de un enlace con el mismo. El proceso de sustitución del archivo propiamente dicho se realiza durante el proceso de confirmación de la transacción actual.

destino-ubicación-datos
Una expresión VARCHAR(200) que especifica un serie de caracteres de longitud variable que contiene un valor de URL completo.

fuente-ubicación-datos
Una expresión VARCHAR que especifica la ubicación de los datos de un archivo en formato URL. Como resultado de una asignación en una sentencia UPDATE o INSERT, el nombre de este archivo se cambia al del archivo al que hace referencia el *destino-ubicación-datos*; los atributos de propiedad y permiso del archivo de destino se conservan.

Existe la restricción de que la *fuente-ubicación-datos* sólo puede ser una de las siguientes:

- Un valor de longitud cero
- Un valor NULL
- El valor de *destino-ubicación-datos* más una serie de sufijo. La serie de sufijo puede tener una longitud máxima de 20 caracteres. Los caracteres de la serie de sufijo deben pertenecer al junto de caracteres URL. Además, la serie no puede contener un carácter "\" bajo el esquema UNC ni el carácter"/" bajo otros esquemas válidos (SQLSTATE 428D1).

serie-comentario
Un valor VARCHAR opcional que contiene un comentario o información adicional sobre la ubicación.

El resultado de la función es un valor DATALINK. Si algún argumento puede ser nulo, el resultado puede ser nulo; si el *destino-ubicación-datos* es nulo, el resultado es el valor nulo.

Si la *fuente-ubicación-datos* es nula, una serie de longitud cero o exactamente igual al *destino-ubicación-datos*, el efecto de DLREPLACECONTENT es el mismo que DLVALUE.

Ejemplo:

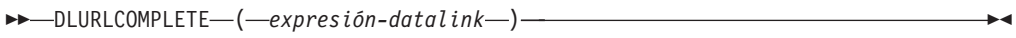
- Sustituya el contenido de un archivo enlazado por otro archivo. Dado un valor DATALINK que se ha insertado en la columna PICT_FILE de la tabla TBLA utilizando la sentencia INSERT siguiente:

```
EXEC SQL INSERT INTO TBLA (PICT_ID, PICT_FILE)
  VALUES(1000, DLVALUE('HTTP://HOSTA.COM/dlfs/image-data/pict1.gif'));
```

Sustituya el contenido de este archivo por otro archivo emitiendo la sentencia SQL UPDATE siguiente:

```
EXEC SQL UPDATE TBLA
  SET PICT_FILE =
    DLREPLACECONTENT('HTTP://HOSTA.COM/dlfs/image-data/pict1.gif',
                     'HTTP://HOSTA.COM/dlfs/image-data/pict1.gif.new')
  WHERE PICT_ID = 1000;
```

DLURLCOMPLETE



La función DLURLCOMPLETE devuelve el atributo de ubicación de datos a partir de un valor DATALINK, con un tipo de enlace de URL. Cuando *expresión-datalink* es una columna DATALINK definida con el atributo READ PERMISSION DB, el valor incluye un símbolo de accesos de archivo.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye el comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
  ```

  la siguiente función que realiza una operación con este valor:

  ```
  DLURLCOMPLETE(COLA)
  ```

  devuelve:

  ```
  HTTP://DLFS.ALMADEN.IBM.COM/x/y/**********;a.b
  ```

  donde ********** representa el símbolo de accesos.

DLURLCOMPLETEONLY

El esquema es SYSIBM.

La función DLURLCOMPLETEONLY devuelve el atributo de ubicación de datos a partir de un valor DATALINK, con un tipo de enlace de URL. El valor devuelto *nunca* incluye un símbolo de accesos de archivo.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye un comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se haya insertado en la columna COLA de DATALINK (definido con READ PERMISSION DB) en la tabla TBLA utilizando la función escalar:

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
  ```

  la siguiente función que realiza una operación con este valor:

  ```
  DLURLCOMPLETEONLY(COLA)
  ```

  devuelve:

  ```
  HTTP://DLFS.ALMADEN.IBM.COM/x/y/a.b
  ```

DLURLCOMPLETEWRITE

El esquema es SYSIBM.

La función DLURLCOMPLETEWRITE devuelve el valor de URL completo a partir de un valor DATALINK con un tipo de enlace de URL. Si el valor de DATALINK generado a partir de *expresión-datalink* procede de una columna DATALINK definida con WRITE PERMISSION ADMIN, se incluye un símbolo de escritura en el valor devuelto. El valor devuelto puede utilizarse para localizar y actualizar el archivo enlazado.

Si la columna DATALINK está definida con otra opción de WRITE PERMISSION (no ADMIN) o con NO LINK CONTROL, DLURLCOMPLETEWRITE sólo devuelve el valor del URL sin un símbolo de escritura. Si la referencia de archivo se deriva de una columna DATALINK definida con WRITE PERMISSION FS, no se necesita ningún símbolo para grabar en el archivo, porque el sistema de archivos controla el permiso de escritura; si la referencia de archivo se deriva de una columna DATALINK definida con WRITE PERMISSION BLOCKED, no puede escribirse nada en el archivo.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye un comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se haya insertado en la columna COLA de DATALINK (definido con WRITE PERMISSION ADMIN) en la tabla TBLA utilizando la función escalar:

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
  ```

  la siguiente función que realiza una operación con este valor:

  ```
  DLURLCOMPLETEWRITE(COLA)
  ```

  devuelve:

  ```
  HTTP://DLFS.ALMADEN.IBM.COM/x/y/**********;a.b
  ```

  donde ********** representa el símbolo de escritura. Si COLA no está definido con WRITE PERMISSION ADMIN, el símbolo de escritura no aparecerá.

DLURLPATH

El esquema es SYSIBM.

La función DLURLPATH devuelve la vía de acceso y el nombre de archivo necesarios para acceder a un archivo de un servidor determinado desde un valor DATALINK con un tipoenlace de URL. Cuando *expresión-datalink* es una columna DATALINK definida con el atributo READ PERMISSION DB, el valor incluye un símbolo de accesos de archivo.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye el comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
  ```

  la siguiente función que realiza una operación con este valor:

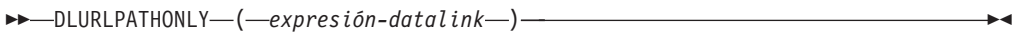
  ```
  DLURLPATH(COLA)
  ```

  devolverá el valor:

  ```
  /x/y/**********;a.b
  ```

  (donde ********** representa el símbolo de accesos)

DLURLPATHONLY



El esquema es SYSIBM.

La función DLURLPATHONLY devuelve la vía de acceso y el nombre de archivo necesarios para acceder a un archivo de un servidor determinado desde un valor DATALINK con un tipo de enlace de URL. El valor devuelto NUNCA incluye un símbolo de accesos de archivo.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye el comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
  ```

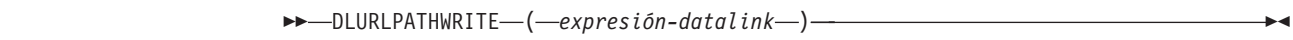
  la siguiente función que realiza una operación con este valor:

  ```
  DLURLPATHONLY(COLA)
  ```

  devolverá el valor:

  ```
  /x/y/a.b
  ```

DLURLPATHWRITE



El esquema es SYSIBM.

La función DLURLPATHWRITE devuelve la vía de acceso y el nombre de archivo necesarios para acceder a un archivo de un servidor determinado desde un valor DATALINK con un tipo de enlace de URL. El valor devuelto incluye un símbolo de escritura si el valor de DATALINK generado a partir de *expresión-datalink* procede de una columna DATALINK definida con WRITE PERMISSION ADMIN.

Si la columna DATALINK está definida con otras opciones de WRITE PERMISSION (no ADMIN) o con NO LINK CONTROL, DLURLPATHWRITE devuelve la vía de acceso y el nombre de archivo sin un símbolo de escritura. Si la referencia de archivo se deriva de una columna DATALINK definida con WRITE PERMISSION FS, no se necesita ningún símbolo para grabar en el archivo, porque el sistema de archivos controla el permiso de escritura; si la referencia de archivo se deriva de una columna DATALINK definida con WRITE PERMISSION BLOCKED, no puede escribirse nada en el archivo.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye un comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se haya insertado en la columna COLA de DATALINK (definido con WRITE PERMISSION ADMIN) en la tabla TBLA utilizando la función escalar:

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
  ```

  la siguiente función que realiza una operación con este valor:

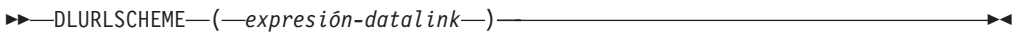
  ```
  DLURLPATHWRITE(COLA)
  ```

  devuelve:

  ```
  /x/y/**********;a.b
  ```

  donde ********** representa el símbolo de escritura. Si COLA no está definido con WRITE PERMISSION ADMIN, el símbolo de escritura no aparecerá.

DLURLSCHEME



El esquema es SYSIBM.

La función DLURLSCHEME devuelve el esquema de un valor DATALINK con un tipoenlace de URL. El valor siempre estará en mayúsculas.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(20). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye el comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
  ```

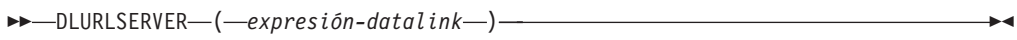
  la siguiente función que realiza una operación con este valor:

  ```
  DLURLSCHEME(COLA)
  ```

  devolverá el valor:

  ```
  HTTP
  ```

DLURLSERVER



El esquema es SYSIBM.

La función DLURLSERVER devuelve el servidor de archivos de un valor DATALINK con un tipoenlace de URL. El valor siempre estará en mayúsculas.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye el comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

  ```
  DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
  ```

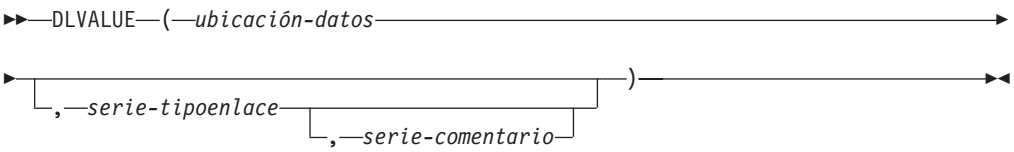
  la siguiente función que realiza una operación con este valor:

  ```
  DLURLSERVER(COLA)
  ```

  devolverá el valor:

  ```
  DLFS.ALMADEN.IBM.COM
  ```

DLVALUE



El esquema es SYSIBM.

La función DLVALUE devuelve un valor DATALINK. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función. Cuando la función está al lado derecho de una cláusula SET dentro de una sentencia UPDATE o está en una cláusula VALUES dentro de una sentencia INSERT, normalmente también crea un enlace con un archivo. No obstante, si sólo se especifica un comentario (en cuyo caso la serie ubicación-datos tiene longitud-cero), el valor DATALINK se crea con atributos de enlace vacíos, por lo que no hay enlace de archivo.

ubicación-datos
Si el tipo de enlace es URL, ésta es una expresión que proporciona una serie de caracteres de longitud variable que contiene un valor de URL completo.

serie-tipoenlace
Una expresión VARCHAR opcional que especifica el tipo de enlace del valor DATALINK. El único valor válido es 'URL' (SQLSTATE 428D1).

serie-comentario
Un valor VARCHAR(254) opcional que proporciona comentarios o información adicional sobre la ubicación. La longitud de *ubicación-datos* más la de *serie-comentarios* no debe sobrepasar los 200 bytes.

El resultado de la función es un valor DATALINK. Si cualquier argumento de la función DLVALUE puede ser nulo, el resultado puede ser nulo; si *ubicación-datos* es nula, el resultado es el valor nulo.

Cuando defina un valor DATALINK mediante esta función, tenga en cuenta la longitud máxima del destino del valor. Por ejemplo, si se define una columna como DATALINK(200), la longitud máxima de *ubicación-datos* más *comentario* suma 200 bytes.

Ejemplo:

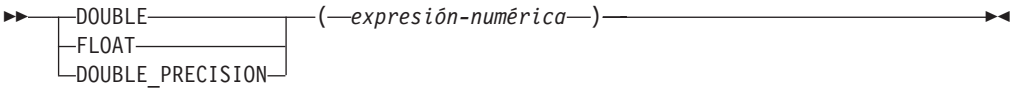
- Inserte una fila en la tabla. Los valores de URL para los dos primeros enlaces se incluyen en las variables denominadas url_article y url_snapshot. La variable denominada url_snapshot_comment contiene un comentario que acompaña el enlace de snapshot. Todavía no existe el enlace de movie, sólo un comentario en la variable denominada url_movie_comment.

```
EXEC SQL
   INSERT INTO HOCKEY_GOALS
            VALUES('Maurice Richard',
                   'Montreal Canadien',
            '?',
                   'Boston Bruins,
                   '1952-04-24',
                   'Winning goal in game 7 of Stanley Cup final',
```

```
DLVALUE(:url_article),
       DLVALUE(:url_snapshot, 'URL', :url_snapshot_comment),
       DLVALUE('', 'URL', :url_movie_comment) );
```

DOUBLE

De numérico a doble :



De serie de caracteres a doble :

El esquema es SYSIBM. Sin embargo, el esquema de DOUBLE(*expresión-serie*) es SYSFUN.

La función DOUBLE devuelve un número de coma flotante correspondiente a:

- un número si el argumento es una expresión numérica
- una representación de serie de caracteres de un número si el argumento es una expresión de serie.

En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

De numérico a doble

expresión-numérica
El argumento es una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

El resultado de la función es un número de coma flotante de precisión doble. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

El resultado es el mismo número que sería si el argumento se hubiese asignado a una columna o variable de coma flotante de precisión doble.

De serie de caracteres a doble

expresión-serie
El argumento puede ser de tipo CHAR o VARCHAR en el formato de una constante numérica. Se ignoran los blancos iniciales y de cola.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

El resultado es el mismo número que sería si la serie se considerase una constante y se asignase a una columna o variable de coma flotante de precisión doble.

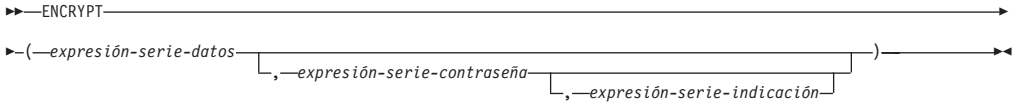
Ejemplo:

Utilizando la tabla EMPLOYEE, busque la proporción de salario y comisiones para los empleados cuya comisión no sea cero. Las columnas implicadas (SALARY y

COMM) tienen tipos de datos DECIMAL. Para eliminar la posibilidad de resultados fuera de rango, se aplica DOUBLE a SALARY para que la división se lleve a cabo en coma flotante:

```
SELECT EMPNO, DOUBLE(SALARY)/COMM
  FROM EMPLOYEE
  WHERE COMM > 0
```

ENCRYPT



El esquema es SYSIBM.

La función ENCRYPT devuelve un valor que es el resultado del cifrado de *expresión-serie-datos*. La contraseña utilizada para el cifrado es el valor de *expresión-serie-contraseña* o el valor de ENCRYPTION PASSWORD (asignado utilizando la sentencia SET ENCRYPTION PASSWORD). En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

expresión-serie-datos
Una expresión que devuelve un valor CHAR o VARCHAR que se debe cifrar. El atributo de longitud para el tipo de datos de *expresión-serie-datos* está limitado a 32663 sin ningún argumento *expresión-serie-indicación* y a 32631 cuando se especifica el argumento *expresión-serie-indicación* (SQLSTATE 42815).

expresión-serie-contraseña
Una expresión que devuelve un valor CHAR o VARCHAR con un mínimo de 6 bytes y no más de 127 bytes (SQLSTATE 428FC). El valor representa la contraseña utilizada para cifrar la *expresión-serie-datos*. Si el valor del argumento de contraseña es nulo o no se proporciona, los datos se cifrarán utilizando el valor de ENCRYPTION PASSWORD, que tiene que haberse establecido para la sesión (SQLSTATE 51039).

expresión-serie-indicación
Una expresión que devuelve un valor CHAR o VARCHAR de un máximo de 32 bytes que ayudará a los propietarios de datos a recordar las contraseñas (por ejemplo, 'Océano' como indicación para recordar 'Pacífico'). Si se proporciona un valor de indicación, la indicación se incorpora en el resultado y puede recuperarse utilizando la función GETHINT. Si este argumento es nulo o no se proporciona, no se incorporará ninguna indicación en el resultado.

El tipo de datos de resultado de la función es VARCHAR FOR BIT DATA.

El atributo de longitud del resultado es:

- Cuando se especifica el parámetro de indicación opcional, el atributo de longitud de los datos no cifrados + 8 bytes + el número de bytes hasta el siguiente límite de 8 bytes + 32 bytes para la longitud de la indicación.
- Sin parámetro de indicación, el atributo de longitud de los datos no cifrados + 8 bytes + el número de bytes hasta el siguiente límite de 8 bytes.

Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

Tenga en cuenta que el resultado cifrado tiene una longitud mayor que la del valor *expresión-serie-datos*. Por consiguiente, al asignar valores cifrados, asegúrese de que el destino se declara con un tamaño suficiente para contener el valor cifrado entero.

Notas:

- ***Algoritmo de cifrado:*** El algoritmo de cifrado interno utilizado es la cifra de bloque RC2 con relleno, la clave secreta de 128 bits se deriva de la contraseña utilizando una conversión de mensaje MD2.
- ***Contraseñas y datos de cifrado*** La gestión de contraseñas es responsabilidad del usuario. Una vez que se han cifrado los datos, sólo se puede utilizar para descifrarlos la contraseña utilizada para cifrarlos (SQLSTATE 428FD). Tenga cuidado al utilizar las variables CHAR para establecer valores de contraseña porque pueden estar rellenadas con espacios en blanco. El resultado cifrado puede contener el terminador nulo y otros caracteres no imprimibles.
- ***Definición de columna de tabla:*** Cuando defina columnas y tipos para que contengan datos cifrados, calcule siempre el atributo de longitud del modo siguiente.

  Para datos cifrados sin indicación:

  Longitud máxima de los datos no cifrados + 8 bytes + el número de bytes hasta el siguiente límite de 8 bytes = longitud de columna de datos cifrados.

  Para datos cifrados con indicación interna:

  Longitud máxima de los datos no cifrados + 8 bytes + el número de bytes hasta el siguiente límite de 8 bytes + 32 bytes para la longitud de la indicación = longitud de columna de datos cifrados.

  Cualquier asignación o cálculo a una longitud inferior a la longitud de datos sugerida puede producir un descifrado anómalo en el futuro y hacer que se pierdan datos. Los espacios en blanco son valores de datos cifrados válidos que se pueden truncar al almacenarse en una columna que es demasiado pequeña.

  Ejemplos de cálculos de longitud de columna:

```
Longitud máxima de datos no cifrados           6 bytes
8 bytes                                        8 bytes
Número de bytes hasta el siguiente límite de
de 8 bytes                                     2 bytes
                                             ---------
Longitud de columna de datos cifrados         16 bytes

Longitud máxima de datos no cifrados          32 bytes
8 bytes                                        8 bytes
Número de bytes hasta el siguiente límite de
8 bytes                                        8 bytes
                                             ---------
Longitud de columna de datos cifrados         48 bytes
```

- ***Administración de datos cifrados:*** Los datos cifrados sólo se pueden descifrar en servidores que soporten las funciones de descifrado que corresponden a la función ENCRYPT. Por lo tanto, el duplicado de columnas con datos cifrados sólo se debe realizar en servidores que soporten la función DECRYPT_BIN o DECRYPT_CHAR.

Ejemplos:

Ejemplo 1: Este ejemplo utiliza el valor de ENCRYPTION PASSWORD para retener la contraseña de cifrado.

```
SET ENCRYPTION PASSWORD = 'Ben123';
  INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832');
```

Ejemplo 2: Este ejemplo pasa explícitamente la contraseña de cifrado.

```
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832','Ben123');
```

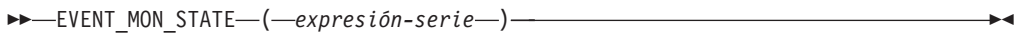
Ejemplo 3: La indicación 'Océano' se almacena para ayudar al usuario a recordar la contraseña de cifrado de 'Pacífico'.

```
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832','Pacífico','Océano');
```

Información relacionada:

- "DECRYPT_BIN y DECRYPT_CHAR" en la página 330
- "GETHINT" en la página 363

EVENT_MON_STATE



El esquema es SYSIBM.

La función EVENT_MON_STATE devuelve el estado actual de un supervisor de sucesos.

El argumento es una expresión de serie con un tipo resultante de CHAR o VARCHAR y un valor que es el nombre de un supervisor de sucesos. Si el supervisor de sucesos nombrado no existe en la tabla del catálogo SYSCAT.EVENTMONITORS, se devolverá SQLSTATE 42704. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado es un entero con uno de los valores siguientes:

0 El supervisor de sucesos está inactivo.

1 El supervisor de sucesos está activo.

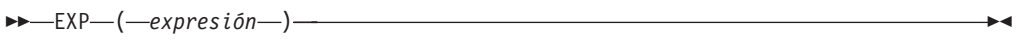
Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

El siguiente ejemplo selecciona todos los supervisores de sucesos definidos e indica si cada uno está activo o inactivo:

```
SELECT EVMONNAME,
  CASE
    WHEN EVENT_MON_STATE(EVMONNAME) = 0 THEN 'Inactive'
    WHEN EVENT_MON_STATE(EVMONNAME) = 1 THEN 'Active'
  END
  FROM SYSCAT.EVENTMONITORS
```

EXP



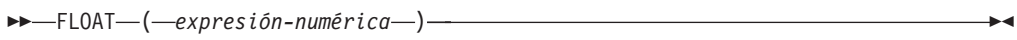
El esquema es SYSFUN.

Devuelve la función exponencial del argumento.

El argumento puede ser de cualquier tipo de datos interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

FLOAT



El esquema es SYSIBM.

La función FLOAT devuelve una representación de coma flotante de un número. FLOAT es sinónimo de DOUBLE.

Información relacionada:

- "DOUBLE" en la página 354

FLOOR

El esquema es SYSIBM. (La versión SYSFUN de la función FLOOR continúa estando disponible).

Devuelve el valor del entero más grande que es menor o igual que el argumento.

El resultado de la función tiene el mismo tipo de datos y el mismo atributo de longitud que el argumento, con la excepción de que la escala es 0 si el argumento es DECIMAL. Por ejemplo, un argumento con un tipo de datos de DECIMAL(5,5) devuelve DECIMAL(5,0).

El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

GETHINT

El esquema es SYSIBM.

La función GETHINT devolverá la indicación de contraseña si se encuentra alguna en *datos-cifrados*. Una indicación de contraseña es una expresión que ayuda a los propietarios de datos a recordar las contraseñas; por ejemplo, 'Océano' como indicación para recordar 'Pacífico'. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

datos-cifrados
Una expresión que devuelve un valor CHAR FOR BIT DATA o VARCHAR FOR BIT DATA que es una serie de datos cifrada completa. La serie de datos se tiene que haber cifrado utilizando la función ENCRYPT (SQLSTATE 428FE).

El resultado de la función es VARCHAR(32). El resultado puede ser nulo; si la función ENCRYPT no ha añadido el parámetro de indicación a los *datos-cifrados* o el primer argumento es nulo, el resultado será el valor nulo.

Ejemplo:

En este ejemplo se almacena la indicación 'Océano' para ayudar al usuario a recordar la contraseña de cifrado 'Pacífico'.

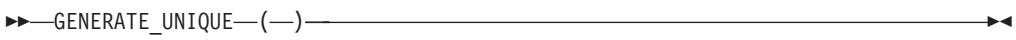
```
INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832', 'Pacífico','Océano');
SELECT GETHINT(SSN)
  FROM EMP;
```

El valor devuelto es 'Océano'.

Información relacionada:

- "DECRYPT_BIN y DECRYPT_CHAR" en la página 330
- "ENCRYPT" en la página 356

GENERATE_UNIQUE



El esquema es SYSIBM.

La función GENERATE_UNIQUE devuelve una serie de caracteres de datos de bits de 13 bytes de longitud (CHAR(13) FOR BIT DATA) que es exclusiva comparada con cualquier otra ejecución de la misma función. (Se utiliza el reloj del sistema para generar la indicación de la Hora Universal Coordinada (UTC) interna junto con el número de partición en la que se ejecuta la función. Los ajustes que retrasan el reloj del sistema real podrían generar valores duplicados). La función se define como no determinista.

No hay ningún argumento para esta función (se han de especificar los paréntesis vacíos).

El resultado de la función es un valor exclusivo que incluye el formato interno de la Hora universal coordinada (UTC) y el número de partición en la que se ha procesado la función. El resultado no puede ser nulo.

El resultado de esta función se puede utilizar para proporcionar valores exclusivos en una tabla. Cada valor sucesivo será mayor que el valor anterior, proporcionando una secuencia que se puede utilizar en una tabla. El valor incluye el número de partición en el que se ha ejecutado la función para que una tabla particionada en múltiples particiones también tenga valores exclusivos en algunas secuencias. La secuencia se basa en la hora en que se ha ejecutado la función.

Esta función difiere de la utilización del registro especial CURRENT TIMESTAMP en que se genera un valor exclusivo para cada fila de una sentencia de inserción de múltiples filas o en una sentencia de inserción con una selección completa.

El valor de indicación de fecha y hora que forma parte del resultado de esta función puede determinarse utilizando la función escalar TIMESTAMP con el resultado de GENERATE_UNIQUE como argumento.

Ejemplos:

- Cree una tabla que incluya una columna que sea exclusiva para cada fila. Llene esta columna utilizando la función GENERATE_UNIQUE. Tenga en cuenta que la columna UNIQUE_ID tiene especificado ″FOR BIT DATA″ para identificar la columna como una serie de caracteres de datos de bits.

  ```
  CREATE TABLE EMP_UPDATE
    (UNIQUE_ID CHAR(13) FOR BIT DATA,
    EMPNO CHAR(6),
    TEXT VARCHAR(1000))
    INSERT INTO EMP_UPDATE
      VALUES (GENERATE_UNIQUE(), '000020', 'Actualizar entrada...'),
      (GENERATE_UNIQUE(), '000050', 'Actualizar entrada...')
  ```

  Esta tabla tendrá un identificador exclusivo para cada fila siempre que la columna UNIQUE_ID se establezca siempre utilizando GENERATE_UNIQUE. Esto se puede realizar introduciendo un activador en la tabla.

```
CREATE TRIGGER EMP_UPDATE_UNIQUE
  NO CASCADE BEFORE INSERT ON EMP_UPDATE
  REFERENCING NEW AS NEW_UPD
  FOR EACH ROW
  SNEW_UPD.UNIQUE_ID = GENERATE_UNIQUE()
```

Con este activador definido, la sentencia INSERT anterior se emitiría sin la primera columna, tal como se indica a continuación.

```
INSERT INTO EMP_UPDATE (EMPNO, TEXT)
  VALUES ('000020', 'Actualizar entrada 1...'),
  ('000050', 'Actualizar entrada 2...')
```

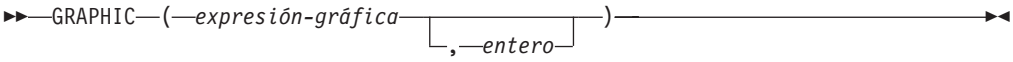
Puede devolverse la indicación de fecha y hora (en UTC) para el momento en que se ha añadido una fila a EMP_UPDATE utilizando:

```
SELECT TIMESTAMP (UNIQUE_ID), EMPNO, TEXT
  FROM EMP_UPDATE
```

Por lo tanto, no hay necesidad de tener una columna de indicación de fecha y hora en la tabla para registrar el momento en que se ha insertado una fila.

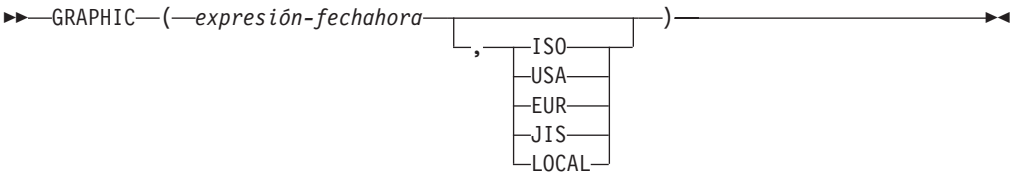
GRAPHIC

De gráfico a gráfico:



De carácter a gráfico:

De fecha y hora a gráfico:



El esquema es SYSIBM.

La función GRAPHIC devuelve una representación de serie gráfica de longitud fija de:

- Una serie gráfica, si el primer argumento es cualquier tipo de serie gráfica
- Un valor de fecha y hora (sólo para bases de datos Unicode), si el primer argumento es una fecha, una hora o una indicación de fecha y hora

En una base de datos Unicode, si un argumento proporcionado es una serie de caracteres, se convertirá a una serie gráfica antes de que se ejecute la función. Cuando la serie de salida se trunca, de forma que el último carácter es un carácter de sustitución elevado, dicho carácter se convierte en un carácter en blanco (X'0020'). No confíe en este comportamiento, porque podría cambiar en los releases futuros.

El resultado de la función es una serie gráfica de longitud fija (tipo de datos GRAPHIC). Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

De gráfico a gráfico

expresión-gráfica
Una expresión que devuelve un valor que es una serie gráfica.

entero
Un valor entero que especifica el atributo de longitud del tipo de datos GRAPHIC resultante. El valor debe estar entre 1 y 127. Si no se especifica ningún valor, el atributo de longitud del resultado es igual al atributo de longitud del primer argumento.

De carácter a gráfico

expresión-caracteres
Una expresión cuyo valor debe ser de tipo de datos de serie de caracteres distinto de LONG VARCHAR o CLOB y cuya longitud máxima es de 16 336 bytes.

El atributo de longitud del resultado es igual al atributo de longitud del argumento.

De fecha y hora a gráfico

expresión-fechahora
Una expresión que sea uno de los tres tipos de datos siguientes:

fecha El resultado es la representación de serie gráfica de la fecha en el formato especificado por el segundo argumento. La longitud del resultado es 10. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

hora El resultado es la representación de serie gráfica de la hora en el formato especificado por el segundo argumento. La longitud del resultado es de 8. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

indicación de fecha y hora
El resultado es la representación de serie gráfica de la indicación de fecha y hora. La longitud del resultado es 26.El segundo argumento no es aplicable y no se debe especificar (SQLSTATE 42815).

La página de códigos de la serie es la página de códigos de la base de datos en el servidor de aplicaciones.

Información relacionada:

• "VARGRAPHIC" en la página 460

HASHEDVALUE

El esquema es SYSIBM.

La función HASHEDVALUE devuelve el índice de mapa de particionamiento de la fila obtenido mediante la aplicación de la función de partición en el valor de clave de particionamiento de la fila. Por ejemplo, si se utiliza en una cláusula SELECT, devuelve el índice de mapa de particionamiento de cada fila de la tabla que se ha utilizado para formar el resultado de la sentencia SELECT.

El índice de mapa de particionamiento devuelto en las tablas y variables de transición procede de los valores de transición actuales de las columnas de claves de particionamiento. Por ejemplo, en un activador BEFORE INSERT, la función devolverá el índice de mapa de particionamiento proyectado teniendo en cuenta los valores actuales de las nuevas variables de transición. No obstante, un activador BEFORE INSERT subsiguiente puede modificar los valores de las columnas de claves de particionamiento. De este modo, el índice de mapa de particionamiento final de la fila cuando se inserte en la base de datos puede ser diferente del valor proyectado.

El argumento debe ser el nombre calificado o no calificado de una columna de una tabla. La columna puede tener cualquier tipo de datos. (Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos como argumento, no es necesario crear signaturas adicionales para dar soporte a los tipos diferenciados definidos por el usuario). Si *nombre-columna* hace referencia a una columna de una vista, la expresión de la vista para la columna debe hacer referencia a una columna de la tabla base principal y la vista debe ser suprimible. Una expresión de tabla anidada o común sigue las mismas reglas que una vista.

La fila específica (y la tabla) para la que la función HASHEDVALUE devuelve el índice de mapa de particionamiento se determina por el contexto de la sentencia de SQL que utiliza la función.

El tipo de datos del resultado es INTEGER en el rango de 0 a 4095. Para una tabla sin clave de particionamiento, el resultado siempre es 0. No se devuelve nunca un valor nulo. Puesto que se devuelve información a nivel de fila, los resultados son los mismos, sin tener en cuenta las columnas que se especifican para la tabla.

La función HASHEDVALUE no puede utilizarse en tablas duplicadas, dentro de restricciones de comprobación ni en la definición de columnas generadas (SQLSTATE 42881).

Para compatibilidad con versiones anteriores a la Versión 8, el nombre de función PARTITION puede sustituirse por HASHEDVALUE.

Ejemplo:

- Liste los números de empleado (EMPNO) de la tabla EMPLOYEE para todas las filas con un índice de mapa de particionamiento de 100.

```
SELECT EMPNO FROM EMPLOYEE
  WHERE HASHEDVALUE(PHONENO) = 100
```

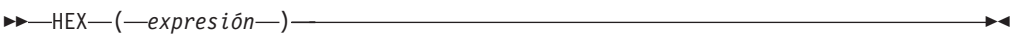
- Anote el número de empleado y el índice de mapa de particionamiento previsto de la nueva fila en una tabla denominada EMPINSERTLOG2 para cualquier inserción de empleados creando un activador BEFORE en la tabla EMPLOYEE.

```
  CREATE TRIGGER EMPINSLOGTRIG2
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AW NEWTABLE
   FOR EACH ROW
   INSERT INTO EMPINSERTLOG2
     VALUES(NEWTABLE.EMPNO, HASHEDVALUE(NEWTABLE.EMPNO))
```

Información relacionada:

- "Sentencia CREATE VIEW" en la publicación *Consulta de SQL, Volumen 2*

HEX



El esquema es SYSIBM.

La función HEX devuelve una representación hexadecimal de un valor como una serie de caracteres.

El argumento puede ser una expresión que sea un valor de cualquier tipo de datos interno, con una longitud máxima de 16.336 bytes.

El resultado de la función es una serie de caracteres. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

La página de códigos es la página de códigos de la base de datos.

El resultado es una serie de dígitos hexadecimales. Los dos primeros bytes representan el primer byte del argumento, los dos siguientes el segundo byte del argumento, etcétera. Si el argumento es un valor de indicación de fecha y hora o un valor numérico el resultado es la representación hexadecimal del formato interno del argumento. La representación hexadecimal que se devuelve puede ser diferente según el servidor de aplicaciones donde se ejecuta la función. Los casos en que las diferencias pueden ser evidentes son:

- Los argumentos de serie de caracteres cuando se ejecuta la función HEX en un cliente ASCII con un servidor EBCDIC o en un cliente EBCDIC con un servidor ASCII.
- Los argumentos numéricos (en algunos casos) cuando se ejecuta la función HEX donde los sistemas cliente y servidor tienen distintas clasificaciones de bytes para los valores numéricos.

El tipo y la longitud del resultado varían basándose en el tipo y la longitud de los argumentos de serie de caracteres.

- Serie de caracteres
  - Longitud fija no mayor que 127
    - El resultado es una serie de caracteres de longitud fija el doble de la longitud definida del argumento.
  - Longitud fija mayor que 127
    - El resultado es una serie de caracteres de longitud variable el doble de la longitud definida del argumento.
  - Longitud variable
    - El resultado es una serie de caracteres de longitud variable con una longitud máxima el doble de la longitud máxima definida del argumento.
- Serie gráfica
  - Longitud fija no mayor que 63
    - El resultado es una serie de caracteres de longitud fija cuatro veces la longitud definida del argumento.
- Longitud fija mayor que 63
  - El resultado es una serie de caracteres de longitud variable cuatro veces la longitud definida del argumento.

- Longitud variable
  - El resultado es una serie de caracteres de longitud variable con una longitud máxima cuatro veces la longitud máxima definida del argumento.

Ejemplos:

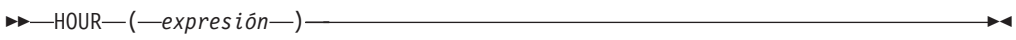
Supongamos que utiliza un servidor de aplicaciones DB2 para AIX en los ejemplos siguientes.

- Utilizando la tabla DEPARTMENT establezca la variable del lenguaje principal HEX_MGRNO (char(12)) en la representación hexadecimal del número del director (MGRNO) para el departamento 'PLANNING' (DEPTNAME).

```
SELECT HEX(MGRNO)
 INTO :HEX_MGRNO
 FROM DEPARTMENT
  WHERE DEPTNAME = 'PLANNING'
```

  HEX_MGRNO se establecerá en '303030303230' cuando se utilice la tabla de ejemplo (el valor de caracteres es '000020').
- Supongamos que COL_1 es una columna con un tipo de datos de char(1) y un valor de 'B'. La representación hexadecimal de la letra 'B' es X'42'. HEX(COL_1) devuelve una serie de dos caracteres '42'.
- Supongamos que COL_3 es una columna con un tipo de datos de decimal(6,2) y un valor de 40,1. Una serie de ocho caracteres '0004010C' es el resultado de aplicar la función HEX a la representación interna del valor decimal 40,1.

HOUR



El esquema es SYSIBM.

La función HOUR devuelve la parte correspondiente a la hora de un valor.

El argumento debe ser una hora, una indicación de fecha y hora, una duración de la hora, una duración de la indicación de fecha y hora o una representación de serie de caracteres válida de una hora o de una fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

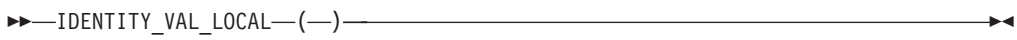
- Si el argumento es una hora, una indicación de fecha y hora o una representación de serie válida de una hora o de una fecha y hora:
  - El resultado es la parte correspondiente a la hora del valor, que es un entero entre 0 y 24.
- Si el argumento es una duración de hora o una duración de indicación de fecha y hora:
  - El resultado es la parte correspondiente a la hora del valor, que es un entero entre –99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplo:

Utilizando la tabla de ejemplo CL_SCHED, seleccione todas las clases que empiezan por la tarde.

```
SELECT * FROM CL_SCHED
 WHERE HOUR(STARTING) BETWEEN 12 AND 17
```

IDENTITY_VAL_LOCAL



El esquema es SYSIBM.

La función IDENTITY_VAL_LOCAL es una función no determinista que devuelve el valor asignado más recientemente de una columna de identidad, donde la asignación se ha producido como resultado de una sentencia de inserción (INSERT) de fila individual utilizando una cláusula VALUES. La función no tiene parámetros de entrada.

El resultado es un DECIMAL(31,0), independientemente del tipo de datos real de la columna de identidad correspondiente.

El valor devuelto por la función es el valor asignado a la columna de identidad de la tabla identificada en la sentencia INSERT de fila individual más reciente. La sentencia INSERT debe contener una cláusula VALUES en una tabla que contenga una columna de identidad. La sentencia INSERT también debe ejecutarse al mismo nivel; es decir, el valor debe estar disponible localmente en el nivel al que se ha asignado, hasta que se sustituya por el siguiente valor asignado. (Se inicia un nivel nuevo cada vez que se invoca un activador o una rutina).

El valor asignado es un valor proporcionado por el usuario (si la columna de identidad está definida como GENERATED BY DEFAULT) o un valor de identidad generado por DB2.

La función devuelve un valor nulo si no se ha emitido una sentencia INSERT de fila individual con una cláusula VALUES en el nivel de proceso actual para una tabla que contiene una columna de identidad.

El resultado de la función no queda afectado por lo siguiente:

- Una sentencia INSERT de fila individual con una cláusula VALUES para una tabla sin columna de identidad.
- Una sentencia INSERT de múltiples filas con una cláusula VALUES.
- Una sentencia INSERT con una selección completa.
- Una sentencia ROLLBACK TO SAVEPOINT.

Notas:

- Las expresiones de la cláusula VALUES de una sentencia INSERT se evalúan antes que las asignaciones para las columnas de destino de la sentencia INSERT. Por consiguiente, una invocación de una función IDENTITY_VAL_LOCAL en la cláusula VALUES de una sentencia INSERT utilizará el valor asignado más recientemente de una columna de identidad de una sentencia INSERT anterior. La función devuelve el valor nulo si no se ha ejecutado ninguna sentencia INSERT de fila individual anterior con una cláusula VALUES para una tabla que contiene una columna de identidad dentro del mismo nivel que la función IDENTITY_VAL_LOCAL.
- El valor de columna de identidad de la tabla para la que se define el activador puede determinarse dentro de un activador, haciendo referencia a la variable de transición activador para la columna de identidad.

- El resultado de la invocación de la función IDENTITY_VAL_LOCAL desde dentro de la condición activador de un activador de inserción es un valor nulo.
- Es posible que existan múltiples activadores de inserción anteriores o posteriores para una tabla. En este caso, cada activador se procesa por separado y los valores de identidad asignados por una acción activada no están disponibles para las demás acciones activadas utilizando la función IDENTITY_VAL_LOCAL. Esto es válido incluso aunque las múltiples acciones activadas estén definidas conceptualmente al mismo nivel.
- Generalmente no es recomendable utilizar la función IDENTITY_VAL_LOCAL en el cuerpo de un activador anterior (before) de inserción. El resultado de la invocación de la función IDENTITY_VAL_LOCAL desde dentro de la acción activada de un activador de inserción anterior es el valor nulo. El valor de la columna de identidad de la tabla para la que se ha definido el activador no se puede obtener invocando la función IDENTITY_VAL_LOCAL en la acción activada de un activador de inserción anterior. Sin embargo, el valor de la columna de identidad puede obtenerse en la acción activada, haciendo referencia a la variable de transición activador para la columna de identidad.
- El resultado de invocar la función IDENTITY_VAL_LOCAL desde la acción activada de un activador posterior (after) de inserción es el valor asignado a una columna de identidad de la tabla identificada en la sentencia INSERT de fila individual más reciente invocada en la misma acción activada que tenía una cláusula VALUES para una tabla que contenía una columna de identidad. (Esto se aplica a los activadores posteriores (after) de inserción FOR EACH ROW y FOR EACH STATEMENT). Si, antes de la invocación de la función IDENTITY_VAL_LOCAL, no se ha ejecutado una sentencia INSERT de fila individual con una cláusula VALUES para una tabla que contiene una columna de identidad dentro de la misma acción activada, la función devolverá un valor nulo.
- Dado que los resultados de la función IDENTITY_VAL_LOCAL no son deterministas, el resultado de una invocación de la función IDENTITY_VAL_LOCAL dentro de la sentencia SELECT de un cursor puede variar para cada sentencia FETCH.
- El valor asignado es el valor realmente asignado a la columna de identidad (es decir, el valor que se devolverá en una sentencia SELECT subsiguiente). Este valor no es necesariamente el valor proporcionado en la cláusula VALUES de la sentencia INSERT o un valor generado por DB2. El valor asignado puede ser un valor especificado en una sentencia de variable de transición SET, dentro del cuerpo de un activador de inserción anterior, para una variable de transición activador asociada con la columna de identidad.
- El valor devuelto por la función es imprevisible después de una operación INSERT de fila individual anómala con una cláusula VALUES en una tabla con una columna de identidad. El valor puede ser el valor que se devolvería de la función si ésta se hubiera invocado antes del INSERT anómalo o puede ser el valor que se asignaría si el INSERT hubiera sido satisfactorio. El valor real devuelto depende del punto de anomalía y, por consiguiente, es imprevisible.

Ejemplos:

Ejemplo 1: Establezca la variable IVAR en el valor asignado a la columna de identidad de la tabla EMPLOYEE. Si esta inserción es la primera en la tabla EMPLOYEE, IVAR tendrá un valor de 1.

```
CREATE TABLE EMPLOYEE
   (EMPNO   INTEGER GENERATED ALWAYS AS IDENTITY,
    NAME CHAR(30),
    SALARY DECIMAL(5,2),
    DEPTNO  SMALLINT)
```

Ejemplo 2: Una función IDENTITY_VAL_LOCAL invocada en una sentencia INSERT devuelve el valor asociado con la sentencia INSERT de fila individual anterior, con una cláusula VALUES para una tabla con una columna de identidad. Para este ejemplo supongamos que existen dos tablas, T1 y T2. T1 y T2 tienen una columna de identidad llamada C1. DB2 genera valores en secuencia, empezando por 1, para la columna C1 de la tabla T1 y valores en secuencia, empezando por 10, para la columna C1 de la tabla T2.

```
CREATE TABLE T1
  (C1 INTEGER GENERATED ALWAYS AS IDENTITY,
   C2 INTEGER)

CREATE TABLE T2
  (C1 DECIMAL(15,0) GENERATED BY DEFAULT AS IDENTITY
    (START WITH 10),
   C2 INTEGER)

INSERT INTO T1 (C2) VALUES (5)

INSERT INTO T1 (C2) VALUES (6)

SELECT * FROM T1
```

lo que da un resultado de:

```
C1           C2
-----------  ----------
          1           5
          2           6
```

y ahora, declarando la función para la variable IVAR:

```
VALUES IDENTITY_VAL_LOCAL() INTO :IVAR
```

En este punto, la función IDENTITY_VAL_LOCAL devolverá un valor de 2 en IVAR, porque ése es el valor que DB2 ha asignado más recientemente. La sentencia INSERT siguiente inserta una fila individual en T2, donde la columna C2 obtiene un valor de 2 de la función IDENTITY_VAL_LOCAL.

```
INSERT INTO T2 (C2) VALUES (IDENTITY_VAL_LOCAL())

SELECT * FROM T2
  WHERE C1 = DECIMAL(IDENTITY_VAL_LOCAL(),15,0)
```

lo que devuelve un resultado de:

```
C1                  C2
------------------  ----------
              10.           2
```

La invocación de la función IDENTITY_VAL_LOCAL después de esta inserción produce un valor de 10, que es el valor generado por DB2 para la columna C1 de T2.

En un entorno anidado que incluya un activador, utilice la función IDENTITY_VAL_LOCAL para recuperar el valor de identidad asignado en un nivel determinado, incluso aunque puedan haber valores de identidad asignados en niveles inferiores. Supongamos que existen tres tablas, EMPLOYEE, EMP_ACT y

ACCT_LOG. Hay un activador de inserción posterior definido en EMPLOYEE que produce inserciones adicionales en las tablas EMP_ACT y ACCT_LOG.

```
CREATE TABLE EMPLOYEE
  (EMPNO SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1000),
   NAME CHAR(30),
   SALARY DECIMAL(5,2),
   DEPTNO SMALLINT);

CREATE TABLE EMP_ACT
  (ACNT_NUM SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1),
   EMPNO SMALLINT);

CREATE TABLE ACCT_LOG
  (ID SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 100),
   ACNT_NUM SMALLINT,
   EMPNO SMALLINT);

CREATE TRIGGER NEW_HIRE
   AFTER INSERT ON EMPLOYEE     REFERENCING NEW AS NEW_EMP
  FOR EACH ROW
  BEGIN ATOMIC

   INSERT INTO EMP_ACT (EMPNO)
      VALUES (NEW_EMP.EMPNO);
      INSERT INTO ACCT_LOG (ACNT_NUM EMPNO)
      VALUES (IDENTITY_VAL_LOCAL(), NEW_EMP.EMPNO);
        END
```

La primera sentencia INSERT activada inserta una fila en la tabla EMP_ACT. Esta sentencia INSERT utiliza una variable de transición activador para la columna EMPNO de la tabla EMPLOYEE, para indicar que el valor de identidad para la columna EMPNO de la tabla EMPLOYEE debe copiarse en la columna EMPNO de la tabla EMP_ACT. La función IDENTITY_VAL_LOCAL no se ha podido utilizar para obtener el valor asignado a la columna EMPNO de la tabla EMPLOYEE. Esto se debe a que no se ha emitido una sentencia INSERT en este nivel del anidamiento y, por lo tanto, si se hubiera invocado la función IDENTITY_VAL_LOCAL en la cláusula VALUES de INSERT para EMP_ACT, se hubiera devuelto un valor nulo. Esta sentencia INSERT para la tabla EMP_ACT también hace que se genere un nuevo valor de columna de identidad para la columna ACNT_NUM.

Una segunda sentencia INSERT activada inserta una fila en la tabla ACCT_LOG. Esta sentencia invoca la función IDENTITY_VAL_LOCAL para indicar que el valor de identidad asignado a la columna ACNT_NUM de la tabla EMP_ACT en la sentencia INSERT anterior de la acción activada debe copiarse en la columna ACNT_NUM de la tabla ACCT_LOG. A la columna EMPNO se le asigna el mismo valor que a la columna EMPNO de la tabla EMPLOYEE.

Desde la aplicación que realiza la invocación (es decir, el nivel en el que se emite INSERT en EMPLOYEE), establezca la variable IVAR en el valor asignado a la columna EMPNO de la tabla EMPLOYEE por la sentencia INSERT original.

```
INSERT INTO EMPLOYEE (NAME, SALARY, DEPTNO)
  VALUES ('Rupert', 989.99, 50);
```

El contenido de las tres tablas después de procesar la sentencia INSERT original y todas las acciones activadas es:

```
   SELECT EMPNO, SUBSTR(NAME,10) AS NAME, SALARY, DEPTNO
     FROM EMPLOYEE;

EMPNO       NAME        SALARY                            DEPTNO
----------- ----------- --------------------------------- -----------
       1000 Rupert                                 989.99          50


   SELECT ACNT_NUM, EMPNO
      FROM EMP_ACT;

ACNT_NUM    EMPNO
----------- -----------
          1        1000


   SELECT * FROM ACCT_LOG;

ID          ACNT_NUM    EMPNO
----------- ----------- -----------
        100           1        1000
```

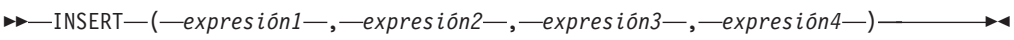
El resultado de la función IDENTITY_VAL_LOCAL es el valor asignado más recientemente para una columna de identidad en el mismo nivel de anidamiento. Después de procesar la sentencia INSERT original y todas las acciones activadas, la función IDENTITY_VAL_LOCAL devuelve un valor de 1000, porque éste es el valor asignado a la columna EMPNO de la tabla EMPLOYEE. La sentencia VALUES siguiente hace que IVAR se establezca en 1000. La inserción en la tabla EMP_ACT (que se ha producido después de la inserción en la tabla EMPLOYEE y a un nivel de anidamiento inferior) no tiene ningún efecto en lo que devuelve esta invocación de la función IDENTITY_VAL_LOCAL.

```
VALUES IDENTITY_VAL_LOCAL() INTO :IVAR;
```

Ejemplos relacionados:

- "fnuse.out -- HOW TO USE BUILT-IN SQL FUNCTIONS (C)"
- "fnuse.sqc -- How to use built-in SQL functions (C)"
- "fnuse.out -- HOW TO USE FUNCTIONS (C++)"
- "fnuse.sqC -- How to use built-in SQL functions (C++)"

INSERT



El esquema es SYSFUN.

Devuelve una serie en la que se han suprimido *expresión3* bytes de *expresión1*, empezando en la *expresión2* y donde se ha insertado la *expresión4* en la *expresión1* empezando en la *expresión2*. Si la longitud de la serie del resultado excede el máximo para el tipo de retorno, se devuelve un error (SQLSTATE 38552).

El primer argumento es una serie de caracteres o un tipo de serie binaria. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función. El segundo argumento y el tercero deben ser un valor numérico con un tipo de datos SMALLINT o INTEGER. Si el primer argumento es una serie de caracteres, entonces el cuarto argumento también ha de ser una serie de caracteres. Si el primer argumento es una serie binaria, entonces el cuarto argumento también ha de ser una serie binaria. Para un VARCHAR la longitud máxima es de 4.000 bytes y para un CLOB o serie binaria la longitud máxima es de 1.048.576 bytes. Para el primer y cuarto argumentos, CHAR se convierte a VARCHAR y LONG VARCHAR a CLOB(1M); para el segundo y tercer argumentos, SMALLINT se convierte a INTEGER para que lo procese la función.

El resultado se basa en los tipos de argumentos tal como sigue:

- VARCHAR(4000) si el primer argumento y el cuarto son VARCHAR (no exceden de 4.000 bytes) o CHAR
- CLOB(1M) si el primer argumento o el cuarto es CLOB o LONG VARCHAR
- BLOB(1M) si el primer argumento y el cuarto son BLOB.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Suprima un carácter de la palabra 'DINING' e inserte 'VID', empezando en el tercer carácter.

  ```
  VALUES CHAR(INSERT('DINING', 3, 1, 'VID'), 10)
  ```

  Este ejemplo devuelve lo siguiente:

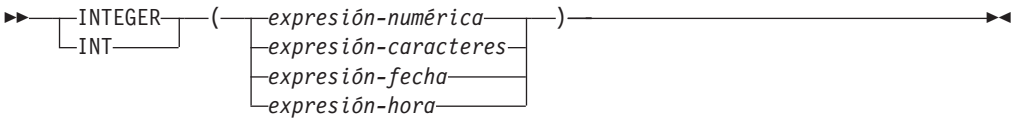
  ```
  1
  ----------
  DIVIDING
  ```

  Tal como se menciona, el resultado de la función INSERT es VARCHAR(4000). En este ejemplo, se ha utilizado la función CHAR para limitar la salida de INSERT a 10 bytes. La ubicación inicial de una serie en particular se puede encontrar utilizando la función LOCATE.

Información relacionada:

- "LOCATE" en la página 387

INTEGER



El esquema es SYSIBM.

La función INTEGER devuelve una representación entera de un número, serie de caracteres, fecha u hora en forma de una constante entera. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

Si el argumento es una *expresión-numérica*, el resultado es el mismo número que sería si el argumento se asignase a una columna o variable de enteros grandes. Si la parte correspondiente a los enteros del argumento no está dentro del rango de enteros, se produce un error. La parte correspondiente a los decimales del argumento se trunca si está presente.

expresión-caracteres
Una expresión que devuelve un valor de serie de caracteres de longitud no mayor que la longitud máxima de una constante de caracteres. Se eliminan los blancos iniciales y de cola y la serie resultante debe ajustarse a las reglas para la formación de una constante de enteros SQL (SQLSTATE 22018). La serie de caracteres no puede ser una serie larga.

Si el argumento es una *expresión-caracteres*, el resultado es el mismo número que sería si la constante de enteros correspondiente se asignase a una columna o variable de enteros grandes.

expresión-fecha
Una expresión que devuelve un valor del tipo de datos DATE. El resultado es un valor INTEGER que representa la fecha como *aaaammdd*.

expresión-hora
Una expresión que devuelve un valor del tipo de datos TIME. El resultado es un valor INTEGER que representa la hora como *hhmmss*.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos:

- Utilizando la tabla EMPLOYEE, seleccione una lista que contenga el salario (SALARY) dividido por el nivel de formación (EDLEVEL). Trunque cualquier decimal en el cálculo. La lista también debe contener los valores utilizados en el cálculo y el número de empleado (EMPNO). La lista debe estar en orden descendente del valor calculado.

```
SELECT INTEGER (SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
  FROM EMPLOYEE
  ORDER BY 1 DESC
```

- Utilizando la tabla EMPLOYEE, seleccione la columna EMPNO en el formato de enteros para procesarla más en la aplicación.

```
SELECT INTEGER(EMPNO) FROM EMPLOYEE
```

- Supongamos que la columna BIRTHDATE (fecha) tiene un valor interno equivalente a '1964-07-20'.

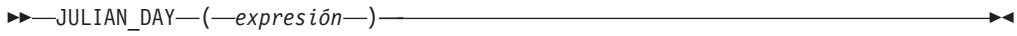
  ```
  INTEGER(BIRTHDATE)
  ```

  da como resultado el valor `19 640 720`.
- Supongamos que la columna STARTTIME (hora) tiene un valor interno equivalente a '12:03:04'.

  ```
  INTEGER(STARTTIME)
  ```

  da como resultado el valor `120 304`.

JULIAN_DAY



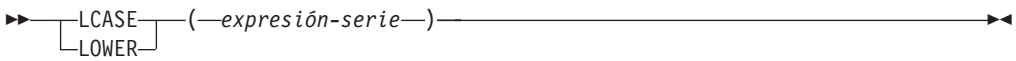
El esquema es SYSFUN.

Devuelve una representación de un valor entero del número de días desde el uno de enero de 4713 A.C. (la fecha de inicio del calendario juliano) hasta el valor de fecha especificado en el argumento.

El argumento debe ser una fecha, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o de una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LCASE o LOWER



El esquema es SYSIBM. (La versión SYSFUN de esta función continúa estando disponible con el soporte para los argumentos LONG VARCHAR y CLOB).

La función LCASE o LOWER devuelve una serie en la que todos los caracteres SBCS se han convertido a minúsculas. Es decir, los caracteres de la A a la Z se convertirán en los caracteres de la a a la z, y los caracteres con signos diacríticos se convertirán a sus minúsculas equivalentes, si existen. Por ejemplo, en la página de códigos 850, É se correlaciona con é. Puesto que no se convierten todos los caracteres, LCASE(UCASE(*expresión-serie*)) no devuelve necesariamente el mismo resultado que LCASE(*expresión-serie*).

El argumento debe ser una expresión cuyo valor sea un tipo de datos CHAR o VARCHAR.

El resultado de la función tiene el mismo tipo de datos y el mismo atributo de longitud que el argumento. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

Ejemplo:

Asegúrese de que los caracteres del valor de la columna JOB de la tabla EMPLOYEE se devuelvan en minúsculas.

```
SELECT LCASE(JOB)
  FROM EMPLOYEE
  WHERE EMPNO = '000020';
```

El resultado es el valor 'director'.

Información relacionada:

- "LCASE (esquema SYSFUN)" en la página 383

LCASE (esquema SYSFUN)

El esquema es SYSFUN.

Devuelve una serie en la que todos los caracteres de la A a la Z se han convertido en caracteres de la a a la z (caracteres con signos diacríticos no convertidos). Tenga en cuenta que, por lo tanto, LCASE(UCASE(serie)) no devolverá necesariamente el mismo resultado que LCASE(serie).

El argumento puede ser de cualquier tipo de serie de caracteres interno. Para un VARCHAR la longitud máxima es de 4.000 bytes y para un CLOB la longitud máxima es de 1.048.576 bytes.

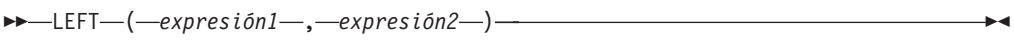
El resultado de la función es:

- VARCHAR(4000) si el argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el argumento es CLOB o LONG VARCHAR

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

LEFT



El esquema es SYSFUN.

Devuelve una serie que consta de los *expresión2* bytes situados más a la izquierda de *expresión1*. El valor *expresión1* se rellena con blancos por la derecha para que la subserie especificada de *expresión1* exista siempre.

El primer argumento es una serie de caracteres o un tipo de serie binaria. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función. Para un VARCHAR la longitud máxima es de 4.000 bytes y para un CLOB o serie binaria la longitud máxima es de 1.048.576 bytes. El segundo argumento debe ser del tipo de datos INTEGER o SMALLINT.

El resultado de la función es:

- VARCHAR(4000) si el argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el argumento es CLOB o LONG VARCHAR
- BLOB(1 M) si el argumento es BLOB.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

LENGTH

El esquema es SYSIBM.

La función LENGTH devuelve la longitud de un valor.

El argumento puede ser una expresión que devuelve un valor de cualquier tipo de datos internos.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

El resultado es la longitud del argumento. La longitud no incluye el byte indicador de nulo de los argumentos de columna que permiten valores nulos. La longitud de las series incluyen blancos pero no incluyen el campo de control de longitud de las series de longitud variable. La longitud de una serie de longitud variable es la longitud real, no la longitud máxima.

La longitud de una serie gráfica es el número de caracteres DBCS. La longitud de todos los demás valores es el número de bytes utilizados para representar el valor:

- 2 para entero pequeño
- 4 para entero grande
- (p/2)+1 para números decimales con la precisión p
- La longitud de la serie para series binarias
- La longitud de la serie para series de caracteres
- 4 para coma flotante de precisión simple
- 8 para coma flotante de precisión doble
- 4 para fecha
- 3 para hora
- 10 para fecha y hora

Ejemplos:

- Supongamos que la variable del lenguaje principal ADDRESS es una serie de caracteres de longitud variable con un valor de '895 Don Mills Road'.

  ```
  LENGTH(:ADDRESS)
  ```

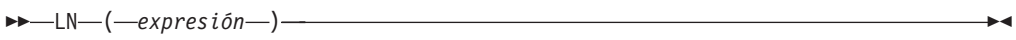
  Devuelve el valor 18.
- Supongamos que START_DATE es una columna de tipo DATE.

  ```
  LENGTH(START_DATE)
  ```

  Devuelve el valor 4.
- Este ejemplo devuelve el valor 10.

  ```
  LENGTH(CHAR(START_DATE, EUR))
  ```

LN



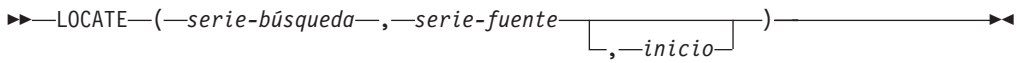
El esquema es SYSFUN.

Devuelve el logaritmo natural del argumento (igual que LOG).

El argumento puede ser de cualquier tipo de datos interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LOCATE



El esquema es SYSFUN.

Devuelve la posición inicial de la primera ocurrencia de la *serie-búsqueda* dentro de la *serie-fuente*. Si se especifica la variable opcional *inicio*, ésta indica la posición de los caracteres en la *serie-fuente* en la que tiene que empezar la búsqueda. La función LOCATE sería entonces equivalente a:

```
POSSTR(SUBSTR(serie-fuente, inicio), serie-búsqueda) + inicio - 1
```

Si no se encuentra la *serie-búsqueda* dentro de la *serie-fuente*, se devuelve el valor 0.

Si el primer argumento es una serie de caracteres, el segundo tendrá que ser una serie de caracteres. Para un VARCHAR la longitud máxima es de 4.000 bytes y para un CLOB la longitud máxima es de 1.048.576 bytes. Si el primer argumento es una serie binaria, el segundo argumento tendrá que ser una serie binaria con una longitud máxima de 1.048.576 bytes. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función. El tercer argumento debe ser INTEGER o SMALLINT.

El resultado de la función es INTEGER. El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplo:

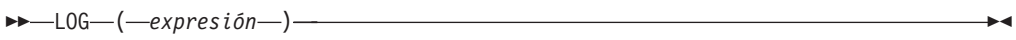
- Busque la ubicación de la letra 'N' (primera ocurrencia) en la palabra 'DINING'.

  ```
  VALUES LOCATE ('N', 'DINING')
  ```

  Este ejemplo devuelve lo siguiente:

  ```
  1
  -----------
  3
  ```

LOG



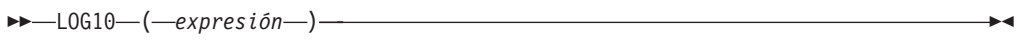
El esquema es SYSFUN.

Devuelve el logaritmo natural del argumento (igual que LN).

El argumento puede ser de cualquier tipo de datos interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LOG10



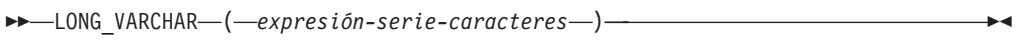
El esquema es SYSFUN.

Devuelve el logaritmo de base 10 del argumento.

El argumento puede ser de cualquier tipo numérico interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LONG_VARCHAR



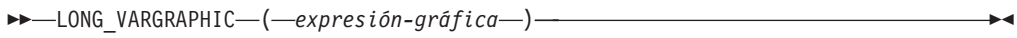
El esquema es SYSIBM.

La función LONG_VARCHAR devuelve una representación LONG VARCHAR de un tipo de datos de serie de caracteres. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

expresión-serie-caracteres
Una *expresión* que devuelve un valor que es una serie de caracteres con una longitud máxima de 32 700 bytes.

El resultado de la función es LONG VARCHAR. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LONG_VARGRAPHIC



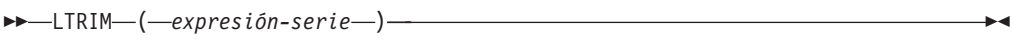
El esquema es SYSIBM.

La función LONG_VARGRAPHIC devuelve una representación LONG VARGRAPHIC de una serie de caracteres de doble byte.

expresión-gráfica
Una *expresión* que devuelve un valor que es una serie gráfica con una longitud máxima de 16 350 caracteres de doble byte.

El resultado de la función es LONG VARGRAPHIC. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LTRIM



El esquema es SYSIBM. (La versión SYSFUN de esta función continúa estando disponible con el soporte para los argumentos LONG VARCHAR y CLOB).

La función LTRIM elimina los blancos del principio de la *expresión-serie*.

El argumento puede ser un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC.

- Si el argumento es una serie gráfica de una base de datos DBCS o EUC, se eliminan los blancos de doble byte iniciales.
- Si el argumento es una serie gráfica de una base de datos Unicode, se eliminan los blancos UCS-2 iniciales.
- De lo contrario, se eliminan los blancos de un solo byte iniciales.

El tipo de datos del resultado de la función es:

- VARCHAR si el tipo de datos de *expresión-serie* es VARCHAR o CHAR
- VARGRAPHIC si el tipo de datos de *expresión-serie* es VARGRAPHIC o GRAPHIC

El parámetro de longitud del tipo devuelto es el mismo que el parámetro de longitud del tipo de datos del argumento.

La longitud real del resultado para las series de caracteres es la longitud de *expresión-serie* menos el número de bytes eliminados debido a caracteres en blanco. La longitud real del resultado para series gráficas es la longitud (en número de caracteres de doble byte) de *expresión-serie* menos el número de caracteres en blanco de doble byte eliminados. Si elimina todos los caracteres se obtiene una serie vacía de longitud variable (longitud de cero).

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

Supongamos que la variable del lenguaje principal HELLO está definida como CHAR(9) y tiene el valor de ' Hola'.

```
VALUES LTRIM(:HELLO)
```

El resultado es 'Hola'.

Información relacionada:

- "LTRIM (esquema SYSFUN)" en la página 393

LTRIM (esquema SYSFUN)

El esquema es SYSFUN.

Devuelve los caracteres del argumento con los blancos iniciales eliminados.

El argumento puede ser de cualquier tipo de serie de caracteres interno. Para un VARCHAR la longitud máxima es de 4.000 bytes y para un CLOB la longitud máxima es de 1.048.576 bytes.

El resultado de la función es:

- VARCHAR(4000) si el argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el argumento es CLOB o LONG VARCHAR.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

MICROSECOND

El esquema es SYSIBM.

La función MICROSECOND devuelve la parte correspondiente a los microsegundos de un valor.

El argumento debe ser una indicación de fecha y hora, una duración de la indicación de fecha y hora o una representación de serie de caracteres válida de una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

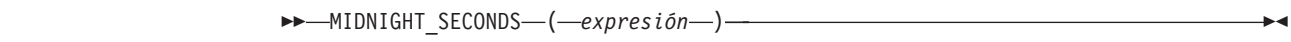
- Si el argumento es una indicación de fecha y hora o una representación de serie válida de una indicación de fecha y hora:
  - El entero está en el rango de 0 a 999 999.
- Si el argumento es una duración:
  - El resultado refleja la parte correspondiente a los microsegundos del valor que es un entero entre –999 999 y 999 999. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplo:

- Supongamos que una tabla TABLEA contiene dos columnas, TS1 y TS2, del tipo TIMESTAMP. Seleccione todas las filas cuya parte correspondiente a los microsegundos de TS1 no sea cero y las partes correspondientes a los segundos de TS1 y TS2 sean idénticas.

```
SELECT * FROM TABLEA
  WHERE MICROSECOND(TS1) <> 0
    AND
  SECOND(TS1) = SECOND(TS2)
```

MIDNIGHT_SECONDS



El esquema es SYSFUN.

Devuelve un valor entero comprendido entre 0 y 86 400 que representa el número de segundos entre medianoche y el valor de la hora especificado en el argumento.

El argumento debe ser una hora, una indicación de fecha y hora o una representación de serie de caracteres válida de una hora o indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos:

- Encuentre el número de segundos entre la medianoche y 00:10:10 y la medianoche y 13:10:10.

  ```
  VALUES (MIDNIGHT_SECONDS('00:10:10'), MIDNIGHT_SECONDS('13:10:10'))
  ```

  Este ejemplo devuelve lo siguiente:

  ```
  1           2
  ----------- -----------
  610         47410
  ```

  Puesto que un minuto es 60 segundos, hay 610 segundos entre la medianoche y la hora especificada. Es lo mismo para el segundo ejemplo. Hay 3600 segundos en una hora y 60 segundos en un minuto, dando como resultado 47410 segundos entre la hora especificada y la medianoche.
- Encuentre el número de segundos entre la medianoche y 24:00:00, y la medianoche y 00:00:00.

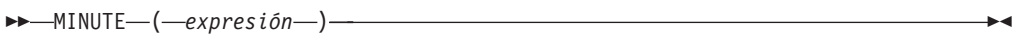
  ```
  VALUES (MIDNIGHT_SECONDS('24:00:00'), MIDNIGHT_SECONDS('00:00:00'))
  ```

  Este ejemplo devuelve lo siguiente:

  ```
  1           2
  ----------- -----------
  86400       0
  ```

  Observe que estos dos valores representan el mismo momento en el tiempo, pero devuelven distintos valores MIDNIGHT_SECONDS.

MINUTE



El esquema es SYSIBM.

La función MINUTE devuelve la parte correspondiente a los minutos de un valor.

El argumento debe ser una hora, una indicación de fecha y hora, una duración de la hora, una duración de la indicación de fecha y hora o una representación de serie de caracteres válida de una hora o de una fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

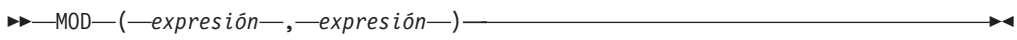
- Si el argumento es una hora, una indicación de fecha y hora o una representación de serie válida de una hora o de una fecha y hora:
  - El resultado es la parte correspondiente a los minutos de un valor, que es un entero entre 0 y 59.
- Si el argumento es una duración de hora o una duración de indicación de fecha y hora:
  - El resultado es la parte correspondiente a los minutos del valor, que es un entero entre –99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplo:

- Utilizando la tabla de ejemplo CL_SCHED, seleccione todas las clases con una duración inferior a 50 minutos.

```
SELECT * FROM CL_SCHED
  WHERE HOUR(ENDING - STARTING) = 0
    AND
  MINUTE(ENDING - STARTING) < 50
```

MOD



El esquema es SYSFUN.

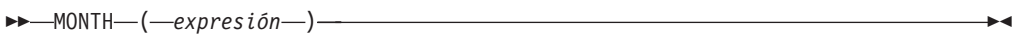
Devuelve el resto del primer argumento dividido por el segundo argumento. El resultado sólo es negativo si el primer argumento es negativo.

El resultado de la función es:

- SMALLINT si ambos argumentos son SMALLINT
- INTEGER si un argumento es INTEGER y el otro es INTEGER o SMALLINT
- BIGINT si un argumento es BIGINT y el otro argumento es BIGINT, INTEGER o SMALLINT.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

MONTH



El esquema es SYSIBM.

La función MONTH devuelve la parte correspondiente al mes de un valor.

El argumento debe ser una fecha, una indicación de fecha y hora, una duración de fecha, una duración de indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una fecha, una indicación de fecha y hora o una representación de serie válida de una fecha o indicación de fecha y hora:
  - El resultado es la parte correspondiente al mes del valor, que es un entero entre 1 y 12.
- Si el argumento es una duración de fecha o duración de indicación de fecha y hora:
  - El resultado es la parte correspondiente al mes del valor, que es un entero entre –99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplo:

- Seleccione todas las filas de la tabla EMPLOYEE de las personas que han nacido (BIRTHDATE) en diciembre (DECEMBER).

```
SELECT * FROM EMPLOYEE
  WHERE MONTH(BIRTHDATE) = 12
```

MONTHNAME

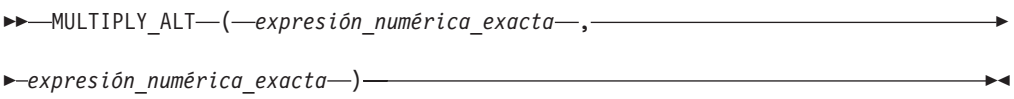
El esquema es SYSFUN.

Devuelve una serie que combina caracteres en mayúsculas y minúsculas que contiene el nombre del mes (por ejemplo, enero) correspondiente a la parte del mes del argumento, según el entorno local en el momento en que se inició la base de datos.

El argumento debe ser una fecha, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o de una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es VARCHAR(100). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

MULTIPLY_ALT



El esquema es SYSIBM.

La función escalar MULTIPLY_ALT devuelve el producto de los dos argumentos como un valor decimal. Se proporciona como alternativa al operador de multiplicación, especialmente cuando la suma de las precisiones de los argumentos excede de 31.

Los argumentos pueden ser cualquier tipo de datos numéricos exactos interno (DECIMAL, BIGINT, INTEGER o SMALLINT).

El resultado de la función es un DECIMAL. La precisión y la escala del resultado se determinan del modo siguiente, utilizando los símbolos *p* y *s* para indicar la precisión y la escala del primer argumento y los símbolos *p'* y *s'* para indicar la precisión y la escala del segundo argumento.

La precisión es MIN(31, *p* + *p'*)

La escala es:

- 0 si la escala de ambos argumentos es 0
- MIN(31, *s* + *s'*) si *p* + *p'* es inferior o igual a 31
- MAX(MIN(3, *s* + *s'*), 31 - (*p* - *s* + *p'* - *s'*)) si *p* + *p'* es superior a 31.

el resultado puede ser nulo si al menos un argumento puede ser nulo o si la base de datos se configura con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si uno de los argumentos es nulo.

Es mejor elegir la función MULTIPLY_ALT que el operador de multiplicación cuando se realizan operaciones aritméticas decimales donde se necesita como mínimo una escala de 3 y la suma de las precisiones excede de 31. En estos casos, se realiza el cálculo interno a fin de evitar desbordamientos. Entonces se asigna el resultado final al tipo de datos de resultado, utilizando el truncamiento donde sea necesario para coincidir con la escala. Tenga en cuenta que el desbordamiento del resultado final sigue siendo posible cuando la escala es 3.

A continuación se proporciona un ejemplo en el que se comparan los tipos de resultado utilizando MULTIPLY_ALT y el operador de multiplicación.

Tipo de argumento 1	Tipo de argumento 2	Resultado utilizando MULTIPLY_ALT	Resultado utilizando el operador de multiplicación
DECIMAL(31,3)	DECIMAL(15,8)	DECIMAL(31,3)	DECIMAL(31,11)
DECIMAL(26,23)	DECIMAL(10,1)	DECIMAL(31,19)	DECIMAL(31,24)
DECIMAL(18,17)	DECIMAL(20,19)	DECIMAL(31,29)	DECIMAL(31,31)
DECIMAL(16,3)	DECIMAL(17,8)	DECIMAL(31,9)	DECIMAL(31,11)
DECIMAL(26,5)	DECIMAL(11,0)	DECIMAL(31,3)	DECIMAL(31,5)
DECIMAL(21,1)	DECIMAL(15,1)	DECIMAL(31,2)	DECIMAL(31,2)

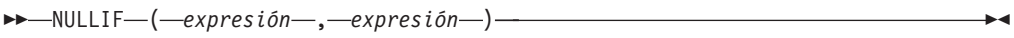
Ejemplo:

Multiplique dos valores donde el tipo de datos del primer argumento es DECIMAL(26,3) y el tipo de datos del segundo argumento es DECIMAL(9,8). El tipo de datos del resultado es DECIMAL(31,7).

```
values multiply_alt(9876543210987654321098 7.654,5.43210987)
1
--------------------------------
 53650467857887529485788 7.5277415
```

Observe que el producto completo de estos dos números es 53650467857887529485788 7.52774154498, pero los 4 últimos dígitos están truncados para coincidir con la escala del tipo de datos de resultado. Si se utiliza el operador de multiplicación con los mismos valores se producirá un desbordamiento aritmético, dado que el tipo de datos de resultado es DECIMAL(31,11) y el valor de resultado tiene 24 dígitos a la izquierda del decimal, pero el tipo de datos de resultado sólo soporta 20 dígitos.

NULLIF



El esquema es SYSIBM.

La función NULLIF devuelve un valor nulo si los argumentos son iguales, de lo contrario, devuelve el valor del primer argumento.

Los argumentos deben poderse comparar. Pueden ser de un tipo de datos interno (que no sea una serie larga ni DATALINK)o de un tipo de datos diferenciado (que no esté basado en una serie larga ni en DATALINK). (Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como esta función acepta cualquier tipo de datos compatible como argumento, no es necesario crear signaturas adicionales para dar soporte a los tipos de datos diferenciados definidos por el usuario). Los atributos del resultado son los atributos del primer argumento.

El resultado de utilizar NULLIF(e1,e2) es igual a utilizar la expresión

```
CASE WHEN e1=e2 THEN NULL ELSE e1 END
```

Observe que cuando e1=e2 se evalúa como desconocido (porque uno o los dos argumentos son NULL), las expresiones CASE lo consideran como no verdadero. Por lo tanto, en esta situación, NULLIF devuelve el valor del primer argumento.

Ejemplo:

- Supongamos que las variables del lenguaje principal PROFIT, CASH y LOSSES tienen tipos de datos DECIMAL con los valores 4500.00, 500.00 y 5000.00 respectivamente:

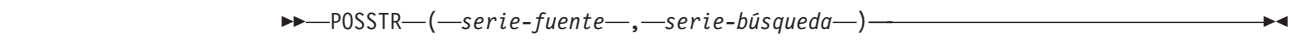
  ```
  NULLIF (:PROFIT + :CASH , :LOSSES )
  ```

  Devuelve un valor nulo.

Información relacionada:

- “Asignaciones y comparaciones” en la página 110

POSSTR



El esquema es SYSIBM.

La función POSSTR devuelve la posición inicial de la primera ocurrencia de una serie (llamada la *serie-búsqueda*) en otra serie (llamada la *serie-fuente*). Los números para la posición de *serie-búsqueda* empiezan en 1 (no en 0).

El resultado de la función es un entero grande. Si alguno de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor nulo.

serie-fuente
Una expresión que especifica la serie fuente en la que se ha de llevar a cabo la búsqueda.

La expresión puede especificarse mediante:

- una constante
- un registro especial
- una variable del lenguaje principal (como por ejemplo, una variable localizadora o una variable de referencia a archivos)
- una función escalar
- un localizador de objeto grande
- un nombre de columna
- una expresión que concatene cualquiera de los elementos anteriores

serie-búsqueda
Una expresión que especifica la serie que se ha de buscar.

La expresión puede especificarse mediante:

- una constante
- un registro especial
- una variable del lenguaje principal
- una función escalar cuyos operandos sean cualquiera de los elementos anteriores
- una expresión que concatene cualquiera de los elementos anteriores

teniendo en cuenta las siguientes restricciones:

- Ningún elemento de la expresión puede ser de tipo LONG VARCHAR, CLOB, LONG VARGRAPHIC o DBCLOB. Además, no puede ser una variable de referencia a archivos BLOB.
- La longitud real de *serie-búsqueda* no puede tener más de 4.000 bytes.

Tanto la *serie-búsqueda* como la *serie-fuente* tienen cero o más posiciones continuas. Si las series son series de caracteres o binarias, una posición es un byte. Si las series son series gráficas, una posición es un carácter gráfico (DBCS).

La función POSSTR acepta las series de datos mixtos. Sin embargo, POSSTR opera sobre la base de un número de bytes estricto, ajeno a los cambios entre caracteres de un solo byte o de múltiples bytes.

Se aplican las siguientes reglas:

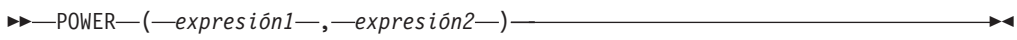
- Los tipos de datos de la *serie-fuente* y la *serie-búsqueda* deben ser compatibles, de lo contrario se genera un error (SQLSTATE 42884).
  - Si *serie-fuente* es una serie de caracteres, entonces *serie-búsqueda* ha de ser una serie de caracteres, pero no una CLOB ni LONG VARCHAR, con una longitud real de 32 672 bytes o menos.
  - Si *serie-fuente* es una serie gráfica, entonces *serie-búsqueda* ha de ser una serie gráfica, pero no una DBCLOB ni LONG VARGRAPHIC, con una longitud real de 16 336 caracteres de doble byte o menos.
  - Si *serie-fuente* es una serie binaria, entonces *serie-búsqueda* ha de ser una serie binaria con una longitud real de 32 672 bytes o menos.
- Si la *serie-búsqueda* tiene una longitud de cero, el resultado que devuelve la función es 1.
- De lo contrario:
  - Si la *serie-fuente* tiene una longitud de cero, el resultado que devuelve la función es cero.
  - De lo contrario:
    - Si el valor de la *serie-búsqueda* es igual a una subserie de longitud idéntica de posiciones continuas del valor de la *serie-fuente*, el resultado devuelto por la función es la posición inicial de la primera de dicha subserie en el valor *serie-fuente*.
    - De lo contrario, el resultado que devuelve la función es 0.

Ejemplo

- Seleccione las columnas RECEIVED y SUBJECT así como la posición inicial de las palabras 'GOOD BEER' de la columna NOTE_TEXT para todas las entradas de la tabla IN_TRAY que contienen estas palabras.

```
SELECT RECEIVED, SUBJECT, POSSTR(NOTE_TEXT, 'GOOD BEER')
 FROM IN_TRAY
 WHERE POSSTR(NOTE_TEXT, 'GOOD BEER') <> 0
```

POWER



El esquema es SYSFUN.

Devuelve el valor de *expresión1* elevado a la potencia de *expresión2*.

Los argumentos pueden ser de cualquier tipo de datos numérico interno. Los argumentos DECIMAL y REAL se convierten a un número de coma flotante de precisión doble.

El resultado de la función es:

- INTEGER si ambos argumentos son INTEGER o SMALLINT
- BIGINT si un argumento es BIGINT y el otro argumento es BIGINT, INTEGER o SMALLINT
- de lo contrario, DOUBLE.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

QUARTER

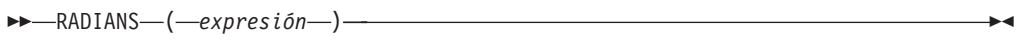
El esquema es SYSFUN.

Devuelve un valor entero comprendido entre 1 y 4 que representa el trimestre del año para la fecha especificada en el argumento.

El argumento debe ser una fecha, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o de una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

RADIANS



El esquema es SYSFUN.

Devuelve el número de radianes convertidos del argumento que se expresa en grados.

El argumento puede ser de cualquier tipo de datos interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

RAISE_ERROR

El esquema es SYSIBM.

La función RAISE_ERROR hace que la sentencia que incluye la función devuelva un error especificando SQLSTATE, SQLCODE -438 y la *serie-diagnóstico*. La función RAISE_ERROR siempre devuelve NULL con un tipo de datos no definido. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

sqlstate
Una serie de caracteres que contiene 5 caracteres exactamente. Debe ser de tipo CHAR definido con una longitud de 5 o un tipo VARCHAR definido con una longitud de 5 o mayor. El valor de *sqlstate* debe seguir las reglas de los SQLSTATE definidos por la aplicación, de la siguiente manera:

- Cada carácter debe pertenecer al conjunto de dígitos (del '0' al '9') o de letras mayúsculas no acentuadas (de la 'A' a la 'Z')
- La clase SQLSTATE (primeros dos caracteres) no puede ser '00', '01' ni '02', pues no son clases de error.
- Si la clase SQLSTATE (primeros dos caracteres) empieza por un carácter de los rangos 0-6 o A-H, la subclase (tres últimos caracteres) debe empezar por una letra del rango I-Z
- Si la clase SQLSTATE (primeros dos caracteres) empieza por un carácter de los rangos 7-9 o I-Z, la subclase (tres últimos caracteres) puede ser un valor cualquiera de 0-9 o A-Z.

Si SQLSTATE no se ajusta a estas reglas se produce un error (SQLSTATE 428B3).

serie-diagnóstico
Una expresión de tipo CHAR o VARCHAR que devuelve una serie de caracteres de un máximo de 70 bytes que describe la condición de error. Si la serie tiene más de 70 bytes de longitud, se truncará.

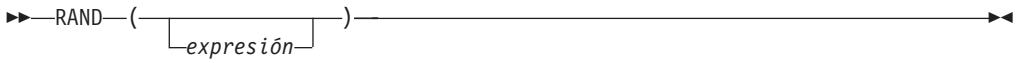
Para utilizar esta función en un contexto en el que las reglas para los tipos de datos del resultado no se aplican (por ejemplo, solo en una lista de selección), se debe utilizar una especificación de conversión (cast) para dar un tipo de datos al valor nulo devuelto. La función RAISE_ERROR resultará más útil en una expresión CASE.

Ejemplo:

Liste los números de empleados y los niveles de formación como, por ejemplo, Postgraduado, Graduado y Diplomado. Si el nivel de formación es mayor que 20, genere un error.

```
SELECT EMPNO,
  CASE WHEN EDUCLVL < 16 THEN 'Diplomado'
       WHEN EDUCLVL < 18 THEN 'Graduado'
       WHEN EDUCLVL < 21 THEN 'Postgraduado'
       ELSE RAISE_ERROR('70001',
         'EDUCLVL tiene un valor mayor que 20')
  END
FROM EMPLOYEE
```

RAND



El esquema es SYSFUN.

La función RAND devuelve un valor de coma flotante comprendido entre 0 y 1.

Si se especifica una expresión, ésta se utiliza como el valor raíz. La expresión debe ser un tipo de datos SMALLINT o INTEGER incorporado con un valor comprendido entre 0 y 2 147 483 647.

El tipo de datos del resultado es de coma flotante de precisión doble. Si el argumento es nulo, el resultado es el valor nulo.

Un valor raíz concreto generará la misma secuencia de números aleatorios para una instancia determinada de una función RAND de una consulta cada vez que se ejecute la consulta. Si no se especifica ningún valor raíz, se genera una secuencia de números aleatorios distinta cada vez que se ejecuta la consulta en la misma sesión. Para generar un conjunto de números aleatorios que varíe de una sesión a otra, debe especificarse un valor raíz aleatorio como, por ejemplo, uno que se base en la hora actual.

RAND es una función no determinante.

REAL

El esquema es SYSIBM.

La función REAL devuelve la representación de coma flotante de precisión simple correspondiente a un número.

El argumento es una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

El resultado de la función es un número de coma flotante de precisión simple. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

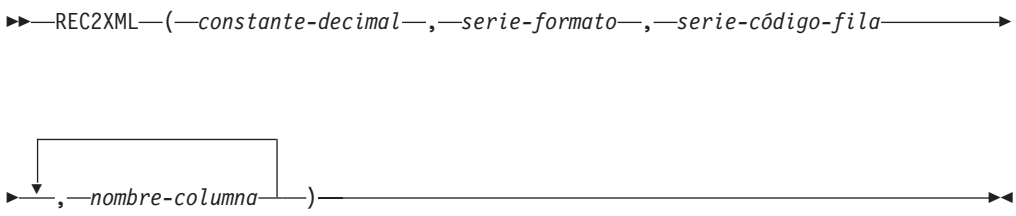
El resultado es el mismo número que sería si el argumento se hubiese asignado a una columna o variable de coma flotante de precisión simple.

Ejemplo:

Utilizando la tabla EMPLOYEE, busque la proporción de salario y comisiones para los empleados cuya comisión no sea cero. Las columnas implicadas (SALARY y COMM) tienen tipos de datos DECIMAL. El resultado se desea en coma flotante de precisión simple. Por consiguiente, se aplica REAL a SALARY para que la división se lleve a cabo en coma flotante (en realidad en precisión doble) y después se aplica REAL a la expresión completa para devolver el resultado en coma flotante de precisión simple.

```
SELECT EMPNO, REAL(REAL(SALARY)/COMM)
  FROM EMPLOYEE
  WHERE COMM > 0
```

REC2XML



El esquema es SYSIBM.

La función REC2XML devuelve una serie formateada codificada en XML que contiene nombres de columna y datos de columna. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

constante-decimal
Factor de expansión para sustituir caracteres de datos de columna. El valor decimal debe ser mayor que 0.0 y menor que o igual a 6.0. (SQLSTATE 42820).

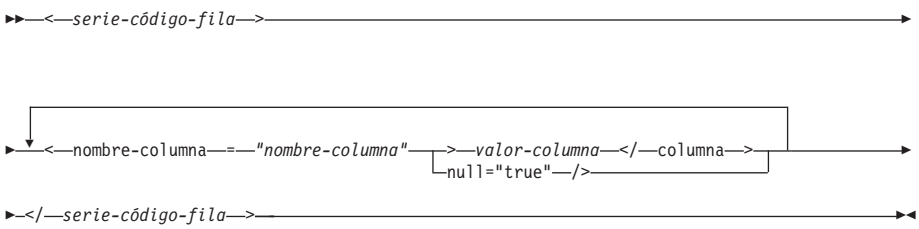
El valor *constante-decimal* se utiliza para calcular la longitud de resultado de la función. Para cada columna con un tipo de datos de tipo carácter, el atributo de longitud de la columna se multiplica por este factor de expansión antes de que se sume a la longitud de resultado.

Para especificar que no haya expansión, utilice un valor de 1.0. Si se especifica un valor menor que 1.0, se reducirá la longitud de resultado calculada. Si la longitud real de la serie de resultado es mayor que la longitud de resultado calculada de la función, se producirá un error (SQLSTATE 22001).

serie-formato
Constante de serie que especifica qué formato debe utilizar la función durante la ejecución.

La *serie-formato* es sensible a las mayúsculas y minúsculas, de modo que los valores siguientes deben especificarse en mayúsculas para que se reconozcan.

COLATTVAL o **COLATTVAL_XML**
Estos formatos devuelven una serie con columnas como valores de atributo.



Los nombres de columna pueden ser valores de atributo XML válidos o pueden no serlo. Para los nombres de columna que no son valores de atributo XML válidos, se realiza la sustitución de caracteres en el nombre de columna antes de incluirlo en la serie de resultado.

Los valores de columna pueden ser nombres de elemento XML válidos o pueden no serlo. Si se especifica la *serie-formato* COLATTVAL, para los nombres de columna que no son valores de elemento XML válidos, se realiza la sustitución de caracteres en el valor de columna antes de incluirlo en la serie de resultado. Si se especifica la *serie-formato* COLATTVAL_XML, no se realiza la sustitución de caracteres en los valores de columna (aunque la sustitución de caracteres se realiza de todas formas en los nombres de columna).

serie-código-fila
Constante de serie que especifica el código utilizado para cada fila. Si se especifica una serie vacía, se supone un valor de 'row'.

Si se especifica una serie de uno o más caracteres en blanco, no aparecerá ninguna *serie-código-fila* inicial o *serie-código-fila* final (incluidos los delimitadores de signo mayor que y menor que) en la serie de resultado.

nombre-columna
Nombre calificado o no calificado de una columna de tabla. La columna debe tener uno de los tipos de datos siguientes (SQLSTATE 42815):

- numérico (SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE)
- serie de caracteres (CHAR, VARCHAR; no se permite una serie de caracteres con un subtipo de BIT DATA)
- fecha y hora (DATE, TIME, TIMESTAMP)
- un tipo definido por el usuario basado en uno de los tipos anteriores

No se puede especificar más de una vez el mismo nombre de columna (SQLSTATE 42734).

El resultado de la función es VARCHAR. La longitud máxima es de 32.672 bytes (SQLSTATE 54006).

Examine la invocación siguiente:

REC2XML (dc, fs, rt, c_1, c_2, ..., c_n)

Si el valor de ″fs″ es ″COLATTVAL″ o ″COLATTVAL_XML″, el resultado será igual que esta expresión:

```
'<' CONCAT rt CONCAT '>' CONCAT y1 CONCAT y2
CONCAT ... CONCAT yn CONCAT '</' CONCAT rt CONCAT '>'
```

donde y_n es equivalente a:

```
'<column name="' CONCAT xvcn CONCAT vn
```

y vn es equivalente a:

```
'">' CONCAT rn CONCAT '</column>'
```

si la columna no es nula y

```
'" null="true"/>'
```

si el valor de columna es nulo.

xvc_n es equivalente a una representación de serie del nombre de columna de c_n, donde cualquier carácter que aparezca en la Tabla 26 en la página 413 se sustituye por la representación correspondiente. Esto asegura que la serie resultante sea un símbolo de valor de elemento o atributo XML válido.

r_n es equivalente a una representación de serie como se indica en la Tabla 25

Tabla 25. Resultado de serie de valores de columna

Tipo de datos de c_n	**r_n**
CHAR, VARCHAR	El valor es una serie. Si la *serie-formato* no termina con los caracteres "_XML", cada carácter de c_n se sustituye por la representación de sustitución correspondiente de la Tabla 26, como se indica. El atributo de longitud es: dc * el atributo de longitud de c_n.
SMALLINT, INTEGER, BIGINT, DECIMAL, NUMERIC, REAL, DOUBLE	El valor es LTRIM(RTRIM(CHAR(c_n))). El atributo de longitud es la longitud de resultado de CHAR(c_n). El carácter decimal es siempre el carácter de punto ('.').
DATE	El valor es CHAR(c_n,ISO). El atributo de longitud es la longitud de resultado de CHAR(c_n,ISO).
TIME	El valor es CHAR(c_n,JIS). El atributo de longitud es la longitud de resultado de CHAR(c_n,JIS)
TIMESTAMP	El valor es CHAR(c_n). El atributo de longitud es la longitud de resultado de CHAR(c_n).

Sustitución de caracteres:

En función del valor especificado para la *serie-formato*, se sustituirán determinados caracteres en los nombres de columna y en los valores de columna para asegurar que los nombres de columna formen valores de atributo XML válidos y que los valores de columna formen valores de elemento XML válidos.

Tabla 26. Sustituciones de caracteres para valores de atributo y valores de elemento XML

Carácter	**Sustitución**
<	&lt;
>	&gt;
"	&quot;
&	&amp;
'	&apos;

Ejemplos:

Nota: REC2XML no inserta caracteres de nueva línea en la salida. Se formatea toda la salida de ejemplo para que se pueda leer.

- Utilizando la tabla DEPARTMENT de la base de datos de ejemplo, formatee la fila de tabla del departamento, excepto las columnas DEPTNAME y LOCATION, para el departamento 'D01' en una serie XML. Dado que los datos no contienen ninguno de los caracteres que necesitan sustituirse, el factor de expansión será 1.0 (sin expansión). Observe también que el valor MGRNO es nulo para esta fila.

```
SELECT REC2XML (1.0, 'COLATTVAL', '', DEPTNO, MGRNO, ADMRDEPT)
 FROM DEPARTMENT
   WHERE DEPTNO = 'D01'
```

Este ejemplo devuelve la serie VARCHAR(117) siguiente:

```
<row>
   <column name="DEPTNO">D01</column>
   <column name="MGRNO" null="true"/>
   <column name="ADMRDEPT">A00</column>
</row>
```

- Una planificación universitaria de 5 días introduce una clase llamada '&43<FIE' en una tabla llamada CL_SCHED, con un formato nuevo para la columna CLASS_CODE. Utilizando la función REC2XML, este ejemplo formatea una serie XML con estos datos de clase nuevos, exceptuando la hora final de clase.

  El atributo de longitud para la llamada REC2XML (vea más abajo) con un factor de expansión de 1.0 será 128 (11 para la actividad general de '<row>' y '</row>', 21 para los nombres de columna, 75 para '<column name=', '>', '</column>' y las comillas dobles, 7 para los datos CLASS_CODE, 6 para los datos DAY y 8 para los datos STARTING). Dado que los caracteres '&' y '<' se sustituirán, no será suficiente un factor de expansión de 1.0. El atributo de longitud de la función necesitará soportar un incremento de 7 a 14 caracteres para los datos CLASS_CODE de formato nuevo.

  Sin embargo, puesto que se sabe que el valor DAY no tendrá nunca más de 1 dígito de longitud, se añaden al total 5 unidades adicionales no utilizadas de longitud. Por lo tanto, la expansión sólo necesita gestionar un incremento de 2. Dado que CLASS_CODE es la única columna de serie de caracteres de la lista de argumentos, éstos son los únicos datos de columna a los que se aplica el factor de expansión. Para obtener un incremento de 2 para la longitud, sería necesario un factor de expansión de 9/7 (1.2857 aproximadamente). Se utilizará un factor de expansión de 1.3.

```
SELECT REC2XML (1.3, 'COLATTVAL', 'record', CLASS_CODE, DAY, STARTING)
   FROM CL_SCHED
   WHERE CLASS_CODE = '&43<FIE'
```

  Este ejemplo devuelve la serie VARCHAR(167) siguiente:

```
<record>
<column name="CLASS_CODE">&amp;43&lt;FIE</column>
   <column name="DAY">5</column>
   <column name="STARTING">06:45:00</column>
</record>
```

- Supongamos que se han añadido filas nuevas a la tabla EMP_RESUME de la base de datos de ejemplo. Las nuevas filas almacenan los resúmenes como series de XML válido. Se utiliza la *serie-formato* COLATTVAL_XML para que no se lleve a cabo la sustitución de caracteres. Ninguno de los resúmenes tiene más de 3500 caracteres de longitud. Se utiliza la consulta siguiente para seleccionar la versión XML de los resúmenes de la tabla EMP_RESUME y formatearla en un fragmento de documento XML.

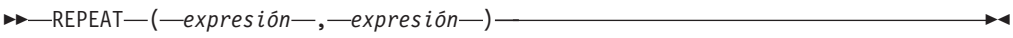
```
SELECT REC2XML (1.0, 'COLATTVAL_XML', 'row', EMPNO, RESUME_XML)
   FROM (SELECT EMPNO, CAST(RESUME AS VARCHAR(3500)) AS RESUME_XML
            FROM EMP_RESUME
            WHERE RESUME_FORMAT = 'XML')
   AS EMP_RESUME_XML
```

  Este ejemplo devuelve una fila para cada empleado que tiene un resumen en formato XML. Cada fila devuelta será una serie con el formato siguiente:

```
<row>
   <column name="EMPNO">{número de empleado}</column>
   <column name="RESUME_XML">{resumen en XML}</column>
</row>
```

Donde ″{número de empleado}″ es el valor EMPNO real para la columna y ″{resumen en XML}″ es el valor de serie de fragmento XML real que constituye el resumen.

REPEAT



El esquema es SYSFUN.

Devuelve una serie de caracteres compuesta por el primer argumento repetido el número de veces especificado por el segundo argumento. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El primer argumento es una serie de caracteres o un tipo de serie binaria. Para un VARCHAR la longitud máxima es de 4.000 bytes y para un CLOB o serie binaria la longitud máxima es de 1.048.576 bytes. El segundo argumento puede ser SMALLINT o INTEGER.

El resultado de la función es:

- VARCHAR(4000) si el primer argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el primer argumento es CLOB o LONG VARCHAR
- BLOB(1 M) si el primer argumento es BLOB.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Liste la frase 'REPITA ESTO' cinco veces.

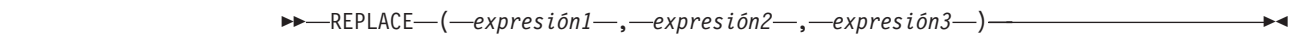
```
VALUES CHAR(REPEAT('REPITA ESTO', 5), 60)
```

  Este ejemplo produce lo siguiente:

```
1
------------------------------------------------------------
REPITA ESTOREPITA ESTOREPITA ESTOREPITA ESTOREPITA ESTO
```

  Tal como se ha mencionado, el resultado de la función REPEAT es VARCHAR(4000). En este ejemplo, se ha utilizado la función CHAR para limitar el resultado de REPEAT a 60 bytes.

REPLACE



El esquema es SYSFUN.

Sustituye todas las ocurrencias de *expresión2* en la *expresión1* por la *expresión3*.

El primer argumento puede ser de cualquier tipo de serie binaria o serie de caracteres interno. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función. Para un VARCHAR la longitud máxima es de 4.000 bytes y para un CLOB o serie binaria la longitud máxima es de 1.048.576 bytes. CHAR se convierte a VARCHAR y LONG VARCHAR se convierte a CLOB(1 M). El tipo del segundo y del tercer argumento es idéntico al del primer argumento.

El resultado de la función es:

- VARCHAR(4000) si el primer, segundo y tercer argumento son VARCHAR o CHAR
- CLOB(1M) si el primer, segundo y tercer argumento son CLOB o LONG VARCHAR
- BLOB(1M) si el primer, segundo y tercer argumento son BLOB.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplo:

Sustituya todas las ocurrencias de la letra 'N' en la palabra 'DINING' por 'VID'.

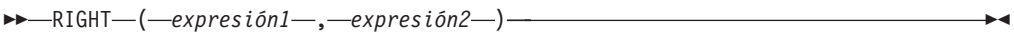
```
VALUES CHAR (REPLACE ('DINING', 'N', 'VID'), 10)
```

Este ejemplo devuelve lo siguiente:

```
1
----------
DIVIDIVIDG
```

Tal como se ha mencionado, el resultado de la función REPLACE es VARCHAR(4000). En este ejemplo, se ha utilizado la función CHAR para limitar el resultado de REPLACE a 10 bytes.

RIGHT



Devuelve una serie que consta de los *expresión2* bytes más a la derecha de *expresión1*. El valor *expresión1* se rellena con blancos por la derecha para que la subserie especificada de *expresión1* exista siempre.

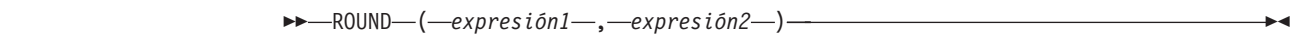
El primer argumento es una serie de caracteres o un tipo de serie binaria. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función. Para un VARCHAR la longitud máxima es de 4.000 bytes y para un CLOB o serie binaria la longitud máxima es de 1.048.576 bytes. El segundo argumento puede ser INTEGER o SMALLINT.

El resultado de la función es:

- VARCHAR(4000) si el primer argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el primer argumento es CLOB o LONG VARCHAR
- BLOB(1 M) si el primer argumento es BLOB.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

ROUND



El esquema es SYSIBM. (La versión SYSFUN de la función ROUND continúa estando disponible).

La función ROUND devuelve *expresión1* redondeada a *expresión2* posiciones a la derecha de la coma decimal si *expresión2* es positiva o a la izquierda de la coma decimal si *expresión2* es cero o negativa.

Si *expresión1* es positiva, un valor de dígito de 5 o mayor es una indicación para redondear al siguiente número positivo más alto. Por ejemplo, ROUND(3.5,0) = 4. Si *expresión1* es negativa, un valor de dígito de 5 o mayor es una indicación para redondear al siguiente número negativo más bajo. Por ejemplo, ROUND(-3.5,0) = -4.

expresión1
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

expresión2
Una expresión que devuelve un entero pequeño o grande. Cuando el valor de *expresión2* no es negativo, especifica redondear a ese número de posiciones a la derecha del separador decimal. Cuando el valor de *expresión2* es negativo, especifica redondear al valor absoluto de *expresión2* posiciones a la izquierda del separador decimal.

Si *expresión2* no es negativa, expresión1 se redondea al valor absoluto del número de *expresión2* de posiciones a la derecha de la coma decimal. Si el valor de *expresión2* es mayor que la escala de *expresión1*, el valor no se modifica excepto si el valor de resultado tiene una precisión que es mayor por 1. Por ejemplo, ROUND(748.58,5) = 748.58 donde la precisión es ahora 6 y la escala sigue siendo 2.

Si *expresión2* es negativa, *expresión1* se redondea al valor absoluto de *expresión2*+1 número de posiciones a la izquierda de la coma decimal.

Si el valor absoluto de una *expresión2* negativa es mayor que el número de dígitos que hay a la izquierda de la coma decimal, el resultado es 0. Por ejemplo, ROUND(748.58,-4) = 0.

El tipo de datos y el atributo de longitud del resultado son iguales que el tipo de datos y el atributo de longitud del primer argumento, excepto en que la precisión se incrementa en uno si la *expresión1* es DECIMAL y la precisión es inferior a 31.

Por ejemplo, un argumento con un tipo de datos de DECIMAL(5,2) da como resultado DECIMAL(6,2). Un argumento con un tipo de datos de DECIMAL(31,2) da como resultado DECIMAL(31,2). La escala es igual que la escala del primer argumento.

Si cualquiera de los argumentos puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES, el resultado puede ser nulo. Si cualquiera de los argumentos es nulo, el resultado es el valor nulo.

Ejemplos:

Calcule el valor de 873.726, redondeado a 2, 1, 0, -1, -2, -3 y -4 posiciones decimales, respectivamente.

```
VALUES (
  ROUND(873.726, 2),
  ROUND(873.726, 1),
  ROUND(873.726, 0),
  ROUND(873.726,-1),
  ROUND(873.726,-2),
  ROUND(873.726,-3),
  ROUND(873.726,-4) )
```

Este ejemplo devuelve:

```
1         2         3         4         5         6         7
--------- --------- --------- --------- --------- --------- ---------
  873.730   873.700   874.000   870.000   900.000  1000.000     0.000
```

Realice el cálculo utilizando los números positivos y negativos.

```
VALUES (
  ROUND(3.5, 0),
  ROUND(3.1, 0),
  ROUNDROUND(-3.1, 0),
  ROUND(-3.5,0) )
```

Este ejemplo devuelve:

```
1    2    3    4
---- ---- ---- ----
 4.0  3.0 -3.0 -4.0
```

RTRIM

El esquema es SYSIBM. (La versión SYSFUN de esta función continúa estando disponible con el soporte para los argumentos LONG VARCHAR y CLOB).

La función RTRIM elimina los blancos del final de la *expresión-serie*.

El argumento puede ser un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC.

- Si el argumento es una serie gráfica de una base de datos DBCS o EUC, se eliminan los blancos de doble byte de cola.
- Si el argumento es una serie gráfica de una base de datos Unicode, se eliminan los blancos UCS-2 de cola.
- De lo contrario, se eliminan los blancos de un solo byte de cola.

El tipo de datos del resultado de la función es:

- VARCHAR si el tipo de datos de *expresión-serie* es VARCHAR o CHAR
- VARGRAPHIC si el tipo de datos de *expresión-serie* es VARGRAPHIC o GRAPHIC

El parámetro de longitud del tipo devuelto es el mismo que el parámetro de longitud del tipo de datos del argumento.

La longitud real del resultado para las series de caracteres es la de *expresión-serie* menos el número de bytes eliminados debido a caracteres en blanco. La longitud real del resultado para series gráficas es la longitud (en número de caracteres de doble byte) de *expresión-serie* menos el número de caracteres en blanco de doble byte eliminados. Si elimina todos los caracteres obtendrá una serie vacía de longitud variable (longitud de cero).

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo: Supongamos que la variable del lenguaje principal HELLO está definida como CHAR(9) y tiene el valor ' Hola'.

```
VALUES RTRIM(:HELLO)
```

El resultado es 'Hola'.

Información relacionada:

- "RTRIM (esquema SYSFUN)" en la página 422

RTRIM (esquema SYSFUN)

El esquema es SYSFUN.

Devuelve los caracteres del argumento con los blancos de cola eliminados.

El argumento puede ser de cualquier tipo de datos de serie de caracteres interno. Para un VARCHAR la longitud máxima es de 4.000 bytes y para un CLOB la longitud máxima es de 1.048.576 bytes.

El resultado de la función es:

- VARCHAR(4000) si el argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el argumento es CLOB o LONG VARCHAR.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo

SECOND

El esquema es SYSIBM.

La función SECOND devuelve la parte correspondiente a los segundos de un valor.

El argumento debe ser una hora, una indicación de fecha y hora, una duración de la hora, una duración de la indicación de fecha y hora o una representación de serie de caracteres válida de una hora o de una fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una hora, una indicación de fecha y hora o una representación de serie válida de una hora o de una fecha y hora:
  - El resultado es la parte correspondiente a los segundos del valor, que es un entero entre 0 y 59.
- Si el argumento es una duración de hora o una duración de indicación de fecha y hora:
  - El resultado es la parte correspondiente a los segundos del valor, que es un entero entre –99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplos:

- Supongamos que la variable del lenguaje principal TIME_DUR (decimal(6,0)) tiene el valor 153045.

  ```
  SECOND(:TIME_DUR)
  ```

  Devuelve el valor 45.
- Supongamos que la columna RECEIVED (indicación de fecha y hora) tiene un valor interno equivalente a 1988-12-25-17.12.30.000000.

  ```
  SECOND(RECEIVED)
  ```

  Devuelve el valor 30.

SIGN

Devuelve un indicador del signo del argumento. Si el argumento es menor que cero, devuelve –1. Si el argumento es igual a cero, devuelve 0. Si el argumento es mayor que cero, devuelve 1.

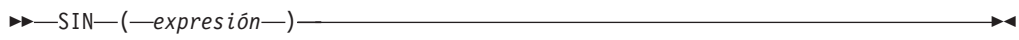
El argumento puede ser de cualquier tipo de datos interno. Los valores DECIMALES y REALES se convierten a números de coma flotante de precisión doble para que los procese la función.

El resultado de la función es:

- SMALLINT si el argumento es SMALLINT
- INTEGER si el argumento es INTEGER
- BIGINT si el argumento es BIGINT
- de lo contrario, DOUBLE.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

SIN



El esquema es SYSIBM. (La versión SYSFUN de la función SIN continúa estando disponible).

Devuelve el seno del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

SINH SINH

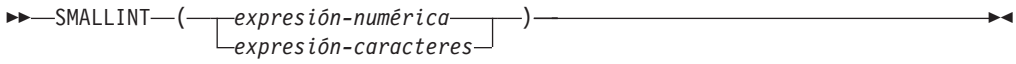
El esquema es SYSIBM.

Devuelve el seno hiperbólico del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

SMALLINT



El esquema es SYSIBM.

La función SMALLINT devuelve una representación de entero pequeño de un número o serie de caracteres en el formato de una constante de enteros pequeños. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

Si el argumento es una *expresión-numérica*, el resultado es el mismo número que sería si el argumento se asignase a una columna o variable de enteros pequeños. Si la parte correspondiente a los enteros del argumento no está dentro del rango de enteros pequeños, se produce un error. La parte correspondiente a los decimales del argumento se trunca si está presente.

expresión-caracteres
Una expresión que devuelve un valor de serie de caracteres de longitud no mayor que la longitud máxima de una constante de caracteres. Se eliminan los blancos iniciales y de cola y la serie resultante debe ajustarse a las reglas para la formación de una constante de enteros SQL (SQLSTATE 22018). Sin embargo, el valor de la constante debe estar en el rango de enteros pequeños (SQLSTATE 22003). La serie de caracteres no puede ser una serie larga.

Si el argumento es una *expresión-caracteres*, el resultado es el mismo número que sería si la constante de enteros correspondiente se asignase a una columna o variable de enteros pequeños.

El resultado de la función es un entero pequeño. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

SOUNDEX

El esquema es SYSFUN.

Devuelve un código de 4 caracteres que representa el sonido de las palabras del argumento. El resultado se puede utilizar para compararlo con el sonido de otras series.

El argumento puede ser una serie de caracteres de tipo CHAR o VARCHAR, cuya longitud no sea mayor que 4.000 bytes. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es CHAR(4). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

La función SOUNDEX es útil para buscar series de las que se conoce el sonido pero no su ortografía exacta. Realiza suposiciones de la manera en que el sonido de las letras y de la combinación de letras puede ayudar a buscar palabras con sonidos similares. La comparación puede realizarse directamente o pasando las series como argumentos a la función DIFFERENCE.

Ejemplo:

Utilizando la tabla EMPLOYEE, busque el EMPNO y el LASTNAME del empleado cuyo apodo suena como 'Loucesy'.

```
SELECT EMPNO, LASTNAME FROM EMPLOYEE
  WHERE SOUNDEX(LASTNAME) = SOUNDEX('Loucesy')
```

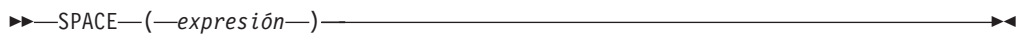
Este ejemplo devuelve lo siguiente:

```
 EMPNO  LASTNAME
------ ---------------
 000110 LUCCHESSI
```

Información relacionada:

- "DIFFERENCE" en la página 334

SPACE



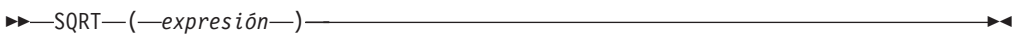
El esquema es SYSFUN.

Devuelve una serie de caracteres que consta de blancos con la longitud especificada por el segundo argumento.

El argumento puede ser SMALLINT o INTEGER.

El resultado de la función es VARCHAR(4000). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

SQRT



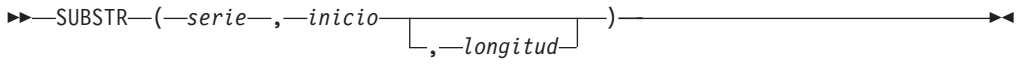
El esquema es SYSFUN.

Devuelve la raíz cuadrada del argumento.

El argumento puede ser de cualquier tipo de datos numérico interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

SUBSTR



La función SUBSTR devuelve una subserie de una serie.

Si la *serie* es una serie de caracteres, el resultado de la función es una serie de caracteres representada en la página de códigos del primer argumento. Si es una serie binaria, el resultado de la función es una serie binaria. Si es una serie gráfica, el resultado de la función es una serie gráfica representada en la página de códigos del primer argumento. Si el primer argumento es una variable de lenguaje principal, la página de códigos del resultado es una página de códigos la base de datos. Si cualquier argumento de la función SUBSTR puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

serie
Una expresión que especifica la serie de la que se deriva el resultado.

Si la *serie* es una serie de caracteres o una serie binaria, una subserie de *serie* es cero o más bytes contiguos de la *serie*. Si la *serie* es una serie gráfica, una subserie de *serie* es cero o más caracteres de doble byte contiguos de *serie*.

inicio
Una expresión que especifica la posición del primer byte del resultado de una serie de caracteres o de una serie binaria o la posición del primer carácter del resultado de una serie gráfica. *inicio* debe ser un entero entre 1 y la longitud o la longitud máxima de la *serie*, según si la *serie* es de longitud fija o de longitud variable (SQLSTATE 22011, si está fuera de rango). Se debe especificar como número de bytes en el contexto de la página de códigos de la base de datos, no de la página de códigos de la aplicación.

longitud
Una expresión que especifica la longitud del resultado. Si se especifica, la *longitud* debe ser un entero binario en el rango de 0 a *n*, donde *n* es igual (el atributo de longitud de la *serie*) − *inicio* + 1 (SQLSTATE 22011, si está fuera de rango).

Si *longitud* se especifica explícitamente, *serie* se rellena por la derecha con el número necesario de caracteres en blanco (de un solo byte para series de caracteres; de doble byte para series gráficas) o caracteres cero hexadecimales (para las series BLOB) para que la subserie especificada de *serie* exista siempre. El valor por omisión para la *longitud* es el número de bytes desde el byte especificado por el *inicio* hasta el último byte de la *serie* en el caso de la serie de caracteres o la serie binaria o el número de caracteres de doble byte del carácter especificado por el *inicio* al último carácter de la *serie* en el caso de una serie gráfica. Sin embargo, si la *serie* es una serie de longitud variable con una longitud menor que el *inicio*, el valor por omisión es cero y el resultado es la serie vacía. Se debe especificar como número de bytes en el contexto de la página de códigos de la base de datos, no de la página de códigos de la aplicación. (Por ejemplo, la columna NAME con un tipo de datos de VARCHAR(18) y un valor de 'MCKNIGHT' dará una serie vacía con `SUBSTR(NAME,10)`.)

La Tabla 27 en la página 432 muestra que el tipo del resultado y la longitud de la función SUBSTR depende del tipo y los atributos de sus entradas.

Tabla 27. Tipo de datos y longitud del resultado de SUBSTR

Tipo de datos del argumento de la serie	Argumento de la longitud	Tipo de datos del resultado
CHAR(A)	constant (*l*<255)	CHAR(*l*)
CHAR(A)	no especificado pero el argumento *inicio* es una constante	CHAR(A-*inicio*+1)
CHAR(A)	no es una constante	VARCHAR(A)
VARCHAR(A)	constant (*l*<255)	CHAR(*l*)
VARCHAR(A)	constant (254<*l*<32673)	VARCHAR(*l*)
VARCHAR(A)	no es una constante o no se especifica	VARCHAR(A)
LONG VARCHAR	constant (*l*<255)	CHAR(*l*)
LONG VARCHAR	constant (254<*l*<4001)	VARCHAR(*l*)
LONG VARCHAR	constant (*l*>4000)	LONG VARCHAR
LONG VARCHAR	no es una constante o no se especifica	LONG VARCHAR
CLOB(A)	constant (*l*)	CLOB(*l*)
CLOB(A)	no es una constante o no se especifica	CLOB(A)
GRAPHIC(A)	constant (*l*<128)	GRAPHIC(*l*)
GRAPHIC(A)	no especificado pero el argumento *inicio* es una constante	GRAPHIC(A-*inicio*+1)
GRAPHIC(A)	no es una constante	VARGRAPHIC(A)
VARGRAPHIC(A)	constant (*l*<128)	GRAPHIC(*l*)
VARGRAPHIC(A)	constant (127<*l*<16337)	VARGRAPHIC(*l*)
VARGRAPHIC(A)	no es una constante	VARGRAPHIC(A)
LONG VARGRAPHIC	constant (*l*<128)	GRAPHIC(*l*)
LONG VARGRAPHIC	constant (127<*l*<2001)	VARGRAPHIC(*l*)
LONG VARGRAPHIC	constant (*l*>2000)	LONG VARGRAPHIC
LONG VARGRAPHIC	no es una constante o no se especifica	LONG VARGRAPHIC
DBCLOB(A)	constant (*l*)	DBCLOB(*l*)
DBCLOB(A)	no es una constante o no se especifica	DBCLOB(A)
BLOB(A)	constant (*l*)	BLOB(*l*)
BLOB(A)	no es una constante o no se especifica	BLOB(A)

Si la *serie* es una serie de longitud fija, la omisión de la *longitud* equivale implícitamente a especificar LENGTH(*serie*) - *inicio* + 1. Si la *serie* es una serie de longitud variable, la omisión de la *longitud* equivale implícitamente a especificar 0 o LENGTH(*serie*) - *inicio* + 1, lo que sea mayor.

Ejemplos:

- Supongamos que la variable del lenguaje principal NAME (VARCHAR(50)) tiene un valor de 'BLUE JAY' y la variable del lenguaje principal SURNAME_POS (int) tiene un valor de 6.

  ```
  SUBSTR(:NAME, :SURNAME_POS)
  ```

  Devuelve el valor 'JAY'

  ```
  SUBSTR(:NAME, :SURNAME_POS,1)
  ```

  Devuelve el valor 'J'.

- Seleccione todas las filas de la tabla PROJECT para las que el nombre del proyecto (PROJNAME) empiece por la palabra 'OPERATION '.

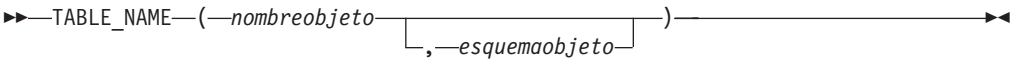
  ```
  SELECT * FROM PROJECT
    WHERE SUBSTR(PROJNAME,1,10) = 'OPERATION '
  ```

  El espacio al final de la constante es necesario como preludio de las palabras iniciales como 'OPERATIONS'.

Notas:

1. En SQL dinámico, *serie*, *inicio* y *longitud* pueden estar representados por un marcador de parámetros (?). Si se utiliza un marcador de parámetros para la *serie*, el tipo de datos del operando será VARCHAR y el operando podrá contener nulos.
2. Aunque no se indique explícitamente en las definiciones del resultado anteriores, se deriva de esta semántica que si la *serie* es una serie de caracteres mixtos de un solo byte y de múltiples bytes, el resultado puede contener fragmentos de caracteres de múltiples bytes, según los valores de *inicio* y *longitud*. Es decir, posiblemente el resultado podría empezar en el segundo byte de un carácter de doble byte y/o finalizar en el primer byte de un carácter de doble byte. La función SUBSTR no detecta dichos fragmentos, ni proporciona ningún proceso especial si se producen.

TABLE_NAME



El esquema es SYSIBM.

La función TABLE_NAME devuelve un nombre no calificado del objeto encontrado después de que se haya resuelto cualquier cadena de seudónimos. El *nombreobjeto* especificado (y el *esquemaobjeto*) se utilizan como el punto de inicio de la resolución. Si el punto de inicio no hace referencia a un seudónimo, se devuelve el nombre no calificado del punto de inicio. El nombre resultante puede ser de una tabla, de una vista o de un objeto no definido. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

nombreobjeto
Una expresión de caracteres que representa el nombre no calificado (normalmente de un seudónimo existente) que se ha de resolver. El tipo de datos de *nombreobjeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor 129 caracteres.

esquemaobjeto
Una expresión de caracteres que representa el esquema utilizado para calificar el valor del *nombreobjeto* suministrado antes de la resolución. El tipo de datos de *esquemaobjeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor 129 caracteres.

Si no se suministra el *esquemaobjeto*, se utiliza el esquema por omisión para el calificador.

El tipo de datos del resultado de la función es VARCHAR(128). Si *nombreobjeto* puede ser nulo, el resultado puede ser nulo; si *nombreobjeto* es nulo, el resultado es el valor nulo. Si *esquemaobjeto* es el valor nulo, se utiliza el nombre de esquema por omisión. El resultado es la serie de caracteres que representa un nombre no calificado. El nombre del resultado puede representar uno de los siguientes elementos:

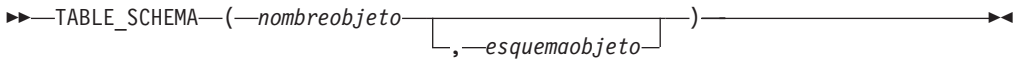
tabla El valor para el *nombreobjeto* era un nombre de tabla (se devuelve el valor de entrada) o un seudónimo que se ha resuelto en la tabla cuyo nombre se devuelve.

vista El valor para el *nombreobjeto* era un nombre de vista (se devuelve el valor de entrada) o un seudónimo que se ha resuelto en la vista cuyo nombre se devuelve.

objeto no definido
El valor para el *nombreobjeto* era un objeto no definido (se devuelve el valor de entrada) o un seudónimo que se ha resuelto en el objeto no definido cuyo nombre se devuelve.

Por lo tanto, si se da un valor no nulo a esta función, siempre se devuelve un valor, incluso si no existe ningún objeto con el nombre del resultado.

TABLE_SCHEMA



El esquema es SYSIBM.

La función TABLE_SCHEMA devuelve el nombre de esquema del objeto encontrado después de que se haya resuelto cualquier cadena de seudónimos. El *nombreobjeto* especificado (y el *esquemaobjeto*) se utilizan como el punto de inicio de la resolución. Si el punto de inicio no hace referencia a un seudónimo, se devuelve el nombre de esquema del punto de inicio. El nombre de esquema resultante puede ser de una tabla, de una vista o de un objeto no definido. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

nombreobjeto
Una expresión de caracteres que representa el nombre no calificado (normalmente de un seudónimo existente) que se ha de resolver. El tipo de datos de *nombreobjeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor 129 caracteres.

esquemaobjeto
Una expresión de caracteres que representa el esquema utilizado para calificar el valor del *nombreobjeto* suministrado antes de la resolución. El tipo de datos de *esquemaobjeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor 129 caracteres.

Si no se suministra el *esquemaobjeto*, se utiliza el esquema por omisión para el calificador.

El tipo de datos del resultado de la función es VARCHAR(128). Si *nombreobjeto* puede ser nulo, el resultado puede ser nulo; si *nombreobjeto* es nulo, el resultado es el valor nulo. Si *esquemaobjeto* es el valor nulo, se utiliza el nombre de esquema por omisión. El resultado es la serie de caracteres que representa un nombre de esquema. El esquema del resultado puede representar el nombre de esquema para uno de los siguientes elementos:

tabla El valor para el *nombreobjeto* era un nombre de tabla (se devuelve la entrada o el valor por omisión de *esquemaobjeto*) o un seudónimo que se ha resuelto en una tabla para la que se devuelve el nombre de esquema.

vista El valor para el *nombreobjeto* era un nombre de vista (se devuelve la entrada o el valor por omisión de *esquemaobjeto*) o un seudónimo que se ha resuelto en una vista para la que se devuelve el nombre de esquema.

objeto no definido
El valor para el *nombreobjeto* era un objeto no definido (se devuelve la entrada o el valor por omisión de *esquemaobjeto*) o un seudónimo que se ha resuelto en un objeto no definido para el que se devuelve el nombre de esquema.

Por lo tanto, si se da a esta función un valor de *nombreobjeto* que no es nulo, siempre se devuelve un valor, incluso si el nombre de objeto con el nombre de esquema del resultado no existe. Por ejemplo, `TABLE_SCHEMA('DEPT', 'PEOPLE')` devuelve 'PEOPLE ' si no se encuentra la entrada del catálogo.

Ejemplos:

- PBIRD intenta seleccionar las estadísticas para una tabla determinada de SYSCAT.TABLES utilizando un seudónimo PBIRD.A1 definido en la tabla HEDGES.T1.

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
 WHERE TABNAME = TABLE_NAME ('A1')
 AND TABSCHEMA = TABLE_SCHEMA ('A1')
```

  Las estadísticas solicitadas para HEDGES.T1 se recuperan del catálogo.

- Seleccione las estadísticas para un objeto llamado HEDGES.X1 de SYSCAT.TABLES utilizando HEDGES.X1. Utilice TABLE_NAME y TABLE_SCHEMA ya que no se conoce si HEDGES.X1 es un seudónimo o una tabla.

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
 WHERE TABNAME = TABLE_NAME ('X1','HEDGES')
 AND TABSCHEMA = TABLE_SCHEMA ('X1','HEDGES')
```

  Suponiendo que HEDGES.X1 sea una tabla, las estadísticas solicitadas para HEDGES.X1 se recuperan del catálogo.

- Seleccione las estadísticas para una tabla determinada de SYSCAT.TABLES utilizando un seudónimo PBIRD.A2 definido en HEDGES.T2 donde HEDGES.T2 no existe.

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
 WHERE TABNAME = TABLE_NAME ('A2','PBIRD')
 AND TABSCHEMA = TABLE_SCHEMA ('A2',PBIRD')
```

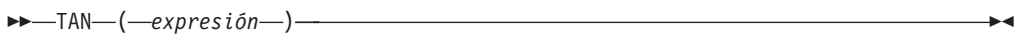
  La sentencia devuelve 0 registros ya que no se encuentra ninguna entrada en SYSCAT.TABLES que coincida donde TABNAME = 'T2' y TABSCHEMA = 'HEDGES'.

- Seleccione el nombre calificado de cada entrada en SYSCAT.TABLES junto con el nombre de referencia final para cualquier entrada de seudónimo.

```
SELECT TABSCHEMA AS SCHEMA,  TABNAME AS NAME,
  TABLE_SCHEMA (BASE_TABNAME, BASE_TABSCHEMA) AS REAL_SCHEMA,
  TABLE_NAME (BASE_TABNAME, BASE_TABSCHEMA) AS REAL_NAME
  FROM SYSCAT.TABLES
```

  La sentencia devuelve el nombre calificado para cada objeto en el catálogo y el nombre de referencia final (después de haberse resuelto el seudónimo) para cualquier entrada de seudónimo. Para todas las entradas que no son seudónimos, BASE_TABNAME y BASE_TABSCHEMA son nulos, por lo tanto las columnas REAL_SCHEMA y REAL_NAME contendrán nulos.

TAN



El esquema es SYSIBM. (La versión SYSFUN de la función TAN continúa estando disponible).

Devuelve la tangente del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos numérico interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

TANH

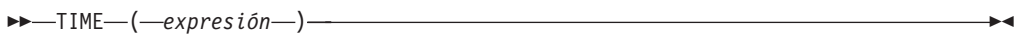
El esquema es SYSIBM.

Devuelve la tangente hiperbólica del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos interno. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

TIME



El esquema es SYSIBM.

La función TIME devuelve una hora de un valor.

El argumento debe ser una hora, una indicación de fecha y hora o una representación de serie válida de una hora o de una indicación de fecha y hora que no sea CLOB, LONG VARCHAR, DBCLOB ni LONG VARGRAPHIC.

Sólo las bases de datos Unicode dan soporte a un argumento que sea una representación de serie gráfica de una hora o una indicación de fecha y hora. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es una hora. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

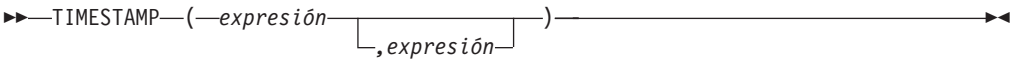
- Si el argumento es una hora:
  - El resultado es dicha hora.
- Si el argumento es una indicación de fecha y hora:
  - El resultado es la parte correspondiente a la hora de la indicación de fecha y hora.
- Si el argumento es una serie:
  - El resultado es la hora representada por la serie.

Ejemplo:

- Seleccione todas las notas de la tabla de ejemplo IN_TRAY que se hayan recibido como mínimo una hora más tarde (de cualquier día) que la hora actual.

```
SELECT * FROM IN_TRAY
  WHERE TIME(RECEIVED) >= CURRENT TIME + 1 HOUR
```

TIMESTAMP



El esquema es SYSIBM.

La función TIMESTAMP devuelve una indicación de fecha y hora a partir de un valor o par de valores.

Sólo las bases de datos Unicode dan soporte a un argumento que es una representación de serie gráfica de una fecha, una hora o una indicación de fecha y hora. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

Las reglas para los argumentos dependen de si se especifica el segundo argumento.

- Si sólo se especifica un argumento:
  - Debe ser una indicación de fecha y hora, una representación de serie válida de una indicación de fecha y hora o una serie de longitud 14 que no sea CLOB, LONG VARCHAR, DBCLOB ni LONG VARGRAPHIC.

    Una serie de longitud 14 debe ser una serie de dígitos que represente una fecha y hora válidas con el formato *aaaaxxddhhmmss*, donde *aaaa* es el año, *xx* es el mes, *dd* es el día, *hh* es la hora, *mm* es el minuto y *ss* es los segundos.
- Si se especifican ambos argumentos:
  - El primer argumento debe ser una fecha o una representación de serie válida de una fecha y el segundo argumento debe ser una hora o una representación de serie válida de una hora.

El resultado de la función es una indicación de fecha y hora. Si el argumento puede ser nulo, el resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen de si se especifica el segundo argumento:

- Si se especifican ambos argumentos:
  - El resultado es una indicación de fecha y hora, en la que el primer argumento especifica la fecha y el segundo argumento especifica la hora. La parte correspondiente a los microsegundos de la indicación de fecha y hora es cero.
- Si sólo se especifica un argumento y es una indicación de fecha y hora:
  - El resultado es esa indicación de fecha y hora.
- Si sólo se especifica un argumento y es una serie:
  - El resultado es la indicación de fecha y hora representada por dicha serie. Si el argumento es una serie de longitud 14, la parte correspondiente a los microsegundos de la indicación de fecha y hora es cero.

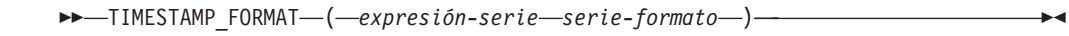
Ejemplo:

- Supongamos que la columna START_DATE (fecha) tiene un valor equivalente a 1988-12-25 y la columna START_TIME (hora) tiene un valor equivalente a 17.12.30.

  ```
  TIMESTAMP(START_DATE, START_TIME)
  ```

  Devuelve el valor '1988-12-25-17.12.30.000000'.

TIMESTAMP_FORMAT



El esquema es SYSIBM.

La función TIMESTAMP_FORMAT devuelve una indicación de fecha y hora a partir de una serie de caracteres que se ha interpretado utilizando una plantilla de caracteres. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

expresión-serie
Una expresión de caracteres que representa un valor de indicación de fecha y hora en el formato especificado por *serie-formato*. (Si *expresión-serie* es un marcador de parámetros sin tipo, se supone que el tipo es VARCHAR con una longitud máxima de 254.) La expresión de serie devuelve un valor CHAR o VARCHAR cuya longitud máxima no es mayor que 254 (SQLSTATE 42815). Los blancos iniciales y de cola se elimina de la *expresión-serie* y la subserie resultante se interpreta como una indicación de fecha y hora utilizando el formato especificado por la *serie-formato*. Los ceros iniciales pueden omitirse de todos los componentes de indicación de fecha y hora, con excepción del año. Es posible utilizar blancos en vez de los ceros iniciales para estos componentes. Por ejemplo, para una serie con el formato `'AAAA-MM-DD HH24:MI:SS'`, todas las series siguientes constituyen una especificación aceptable de las 9 de la mañana del 1 de enero de 2000:

```
'2000-1-01 09:00:00'    (un solo dígito para el mes)
'2000-1-01 09:00:00'    (un solo dígito - precedido por un blanco -
                         para el mes)
'2000-1-1 09:00:00'     (un solo dígito para el mes y el día)
'2000-01-01 9:00:00'    (un solo dígito para la hora)
'2000-01-01 09:0:0'     (un solo dígito para los minutos y los
                         segundos)
'2000- 1- 1 09: 0: 0'   (un solo dígito - precedido por un blanco -
                         para el mes, el día, los minutos y los
                         segundos)
'2000-01-01 09:00:00'   (número máximo de dígitos para cada elemento)
```

serie-formato
Una constante de caracteres que contiene una plantilla para cómo debe interpretarse la expresión de serie como un valor de indicación de fecha y hora. La longitud de la serie de formato no debe ser mayor que 254 (SQLSTATE 42815). Los blancos iniciales y de cola se elimina de la *expresión-formato* y la subserie resultante debe ser una plantilla válida para un valor de indicación de fecha y hora (SQLSTATE 42815). El contenido de *serie-formato* puede especificarse de forma mixta.

Las series de formato válidas son:

```
'AAAA-MM-DD HH24:MI:SS'
```

donde *AAAA* representa un valor de año de 4 dígitos; *MM* representa un valor de mes de 2 dígitos (01-12; enero=01); *DD* representa un valor de día del mes de 2 dígitos (01-31); *HH24* representa un valor de hora del día de 2 dígitos (00-24); Si la hora es 24, el valor de los minutos y de los segundos es cero.); *MI* representa un valor de minutos de 2 dígitos (00-59); y *SS* representa un valor de segundos 2 dígitos (00-59).

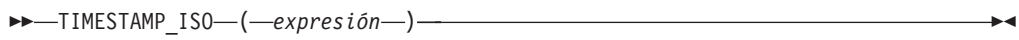
El resultado de la función es una indicación de fecha y hora. Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Inserte una fila en la tabla in_tray con una indicación de fecha y hora de recepción que sea igual a un segundo antes del comienzo del año 2000 (31 de diciembre de 1999 a las 23:59:59).

```
INSERT INTO in_tray (recepción)
  VALUES (TIMESTAMP_FORMAT('1999-12-31 23:59:59',
    'AAAA-MM-DD HH24:MI:SS'))
```

TIMESTAMP_ISO



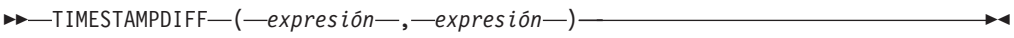
El esquema es SYSFUN.

Devuelve un valor de indicación de fecha y hora basado en un argumento de fecha, de hora o de indicación de fecha y hora. Si el argumento es una fecha, inserta ceros para todos los elementos de hora. Si el argumento es una hora, inserta el valor del registro especial CURRENT DATE para los elementos de fecha y ceros para el elemento de fracción de hora.

El argumento debe ser una fecha, una hora, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha, una hora o una indicación de fecha y hora que no sea ni CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es TIMESTAMP. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

TIMESTAMPDIFF



El esquema es SYSFUN.

Devuelve un número estimado de intervalos del tipo definido por el primer argumento, basándose en la diferencia entre dos indicaciones de la hora.

El primer argumento puede ser INTEGER o SMALLINT. Los valores válidos de intervalo (el primer argumento) son:

1 Fracciones de segundo

2 Segundos

4 Minutos

8 Horas

16 Días

32 Semanas

64 Meses

128 Trimestres

256 Años

El segundo argumento es el resultado de restar dos indicaciones de fecha y hora y convertir el resultado a CHAR(22). En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Se pueden utilizar las suposiciones siguientes al estimar una diferencia:

- Hay 365 días en un año.
- Hay 30 días en un mes.
- Hay 24 horas en un día.
- Hay 60 minutos en una hora.
- Hay 60 segundos en un minuto.

Estas suposiciones se utilizan al convertir la información del segundo argumento, que es una duración de indicación de fecha y hora, al tipo de intervalo especificado en el primer argumento. La estimación que se devuelve puede variar en unos días. Por ejemplo, si se pide el número de días (intervalo 16) para la diferencia entre '1997-03-01-00.00.00' y '1997-02-01-00.00.00', el resultado es 30. Esto es debido a que la diferencia entre las indicaciones de fecha y hora es de 1 mes y se aplica la suposición de que hay 30 días en un mes.

Ejemplo:

El ejemplo siguiente devuelve 4277, el número de minutos entre dos indicaciones de fecha y hora:

```
TIMESTAMPDIFF(4,CHAR(TIMESTAMP('2001-09-29-11.25.42.483219') -
  TIMESTAMP('2001-09-26-12.07.58.065497')))
```

TO_CHAR

El esquema es SYSIBM.

La función TO_CHAR devuelve una representación de caracteres de una indicación de fecha y hora a la que se ha dado formato utilizando una plantilla de caracteres.

TO_CHAR es un sinónimo de VARCHAR_FORMAT.

Información relacionada:

• "VARCHAR_FORMAT" en la página 458

TO_DATE

El esquema es SYSIBM.

La función TO_DATE devuelve una indicación de fecha y hora a partir de una serie de caracteres que se ha interpretado utilizando una plantilla de caracteres.

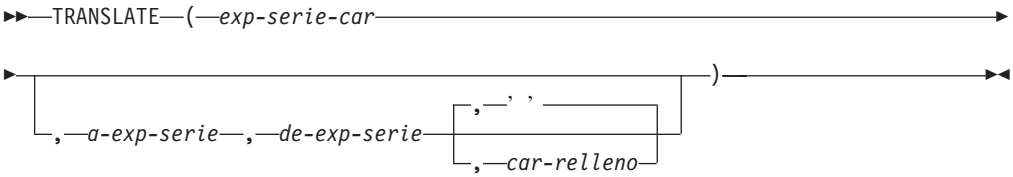
TO_DATE es un sinónimo de TIMESTAMP_FORMAT.

Información relacionada:

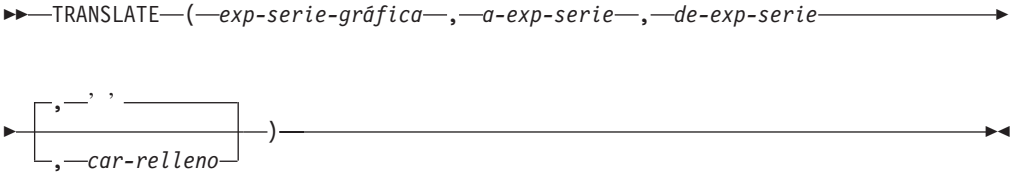
- "TIMESTAMP_FORMAT" en la página 441

TRANSLATE

expresión de serie de caracteres:



expresión de serie gráfica:



El esquema es SYSIBM.

La función TRANSLATE devuelve un valor en el que uno o varios caracteres de una expresión de serie pueden haberse convertido en otros caracteres.

El resultado de la función tiene el mismo tipo de datos y la misma página de códigos que el primer argumento. Si el primer argumento es una variable de lenguaje principal, la página de códigos del resultado es una página de códigos la base de datos. El atributo de longitud del resultado es el mismo que el del primer argumento. Si cualquier expresión especificada puede ser NULL, el resultado puede ser NULL. Si cualquier expresión especificada es NULL, el resultado será NULL.

exp-serie-car o *exp-serie-gráfica*
Una serie que se ha de convertir.

a-exp-serie
Es una serie de caracteres a la cual se convertirán algunos caracteres de *exp-serie-car*.

Si la *a-exp-serie* no está presente y el tipo de datos no es gráfico, todos los caracteres de la *exp-serie-car* se pasarán a mayúsculas, es decir, los caracteres de a-z se convertirán a los caracteres de A-Z y los caracteres con signos diacríticos se convertirán a sus equivalentes en mayúsculas, si existen. Por ejemplo, en la página de códigos 850, é se correlaciona con É pero ÿ no se correlaciona, porque la página de códigos 850 no incluye Ÿ.

de-exp-serie
Es una serie de caracteres que, si se encuentra en la *exp-serie-car*, se convertirá en el carácter correspondiente de la *a-exp-serie*. Si la *de-exp-serie* contiene caracteres duplicados, se utilizará el primero que se encuentre y los demás se pasarán por alto. Si la *a-exp-serie* es más larga que la *de-exp-serie*, se pasarán por alto los caracteres sobrantes. Si la *a-exp-serie* está presente, la *de-exp-serie* también estará presente.

exp-car-relleno
Es un solo carácter que se utilizará para rellenar la *a-exp-serie* si la *a-exp-serie* es más corta que la *de-exp-serie*. La *exp-car-relleno* debe tener un atributo de longitud de uno o se devuelve un error. Si no está presente, se tomará como un blanco de un solo byte.

Los argumentos pueden ser series de tipos de datos CHAR o VARCHAR, o series gráficas de tipo de datos GRAPHIC o VARGRAPHIC. Pueden no tener el tipo de datos LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB ni DBCLOB.

Con la *exp-serie-gráfica*, sólo el *exp-car-relleno* es opcional (si no se proporciona, se tomará el blanco de doble byte) y cada argumento, incluyendo el carácter de relleno, debe ser del tipo de datos gráfico.

El resultado es la serie que aparece después de convertir todos los caracteres de la *exp-serie-car* o la *exp-serie-gráfica* que aparece en la *de-exp-serie* al carácter correspondiente de la *a-exp-serie* o, si no existe ningún carácter correspondiente, al carácter de relleno especificado por la *exp-car-relleno*.

La página de códigos del resultado de TRANSLATE es la misma pagina de códigos que la del primer operando. Para la Versión 8, si el primer operando es una variable del lenguaje principal, la página de códigos del resultado es la página de códigos de la base de datos. Cada uno de los demás operandos se convierte a la página de códigos resultante, a menos que ésta o el primer operando se defina como FOR BIT DATA (en cuyo caso no se realiza ninguna conversión).

Si los argumentos son del tipo de datos CHAR o VARCHAR, los caracteres correspondientes de la *a-exp-serie* y la *de-exp-serie* deben tener el mismo número de bytes. Por ejemplo, no es válido convertir un carácter de un solo byte a un carácter de múltiples bytes o viceversa. Se generará un error si se intenta realizarlo. La *exp-car-relleno* no debe ser el primer byte de un carácter de múltiples bytes válido o se devuelve SQLSTATE 42815. Si la *exp-car-relleno* no está presente, se tomará un blanco de un solo byte.

Si sólo se especifica *exp-serie-car*, los caracteres de un solo byte se convertirán a mayúsculas y los caracteres de múltiples bytes permanecerán sin cambios.

Ejemplos:

- Supongamos que la variable del lenguaje principal SITE (VARCHAR(30)) tiene un valor de 'Hanauma Bay'.

  ```
  TRANSLATE(:SITE)
  ```

  Devuelve el valor 'HANAUMA BAY'.

  ```
  TRANSLATE(:SITE 'j','B')
  ```

  Devuelve el valor 'Hanauma jay'.

  ```
  TRANSLATE(:SITE,'ei','aa')
  ```

  Devuelve el valor 'Heneume Bey'.

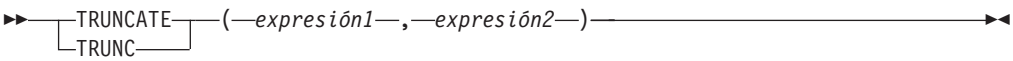
  ```
  TRANSLATE(:SITE,'bA','Bay','%')
  ```

  Devuelve el valor 'HAnAumA bA%'.

  ```
  TRANSLATE(:SITE,'r','Bu')
  ```

  Devuelve el valor 'Hana ma ray'.

TRUNCATE o TRUNC



El esquema es SYSIBM. (La versión SYSFUN de la función TRUNCATE o TRUNC continúa estando disponible).

Devuelve *expresión1* truncada a *expresión2* posiciones a la derecha de la coma decimal si *expresión2* es positiva o a la izquierda de la coma decimal si *expresión2* es cero o negativa.

expresión1
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

expresión2
Una expresión que devuelve un entero pequeño o un entero grande. El valor absoluto del entero especifica el número de posiciones a la derecha de la coma decimal para el resultado si *expresión2* no es negativa o a la izquierda de la coma decimal si *expresión2* es negativa.

Si el valor absoluto de*expresión2* es mayor que el número de dígitos a la izquierda de la coma decimal, el resultado es 0. Por ejemplo:

```
TRUNCATE(748.58,-4) = 0
```

El tipo de datos y el atributo de longitud del resultado de la función tiene el mismo que el tipo de datos y el atributo de longitud del primer argumento.

El resultado puede ser nulo si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si el argumento es nulo.

Ejemplos:

- Utilizando la tabla EMPLOYEE, calcule el salario medio mensual del empleado mejor pagado. Trunque el resultado dos posiciones a la derecha de la coma decimal.

  ```
  SELECT TRUNCATE(MAX(SALARY)/12,2)
     FROM EMPLOYEE;
  ```

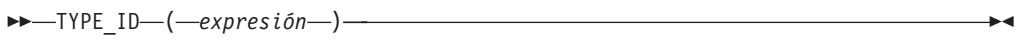
  Como el empleado mejor pagado gana $52750,00 al año, el ejemplo devuelve 4395,83.

- Muestre el número 873,726 truncado 2, 1, 0, -1 y -2 posiciones decimales, respectivamente.

  ```
  VALUES (
    TRUNC(873.726,2),
    TRUNC(873.726,1),
    TRUNC(873.726,0),
    TRUNC(873.726,-1),
    TRUNC(873.726,-2),
    TRUNC(873.726,-3) );
  ```

  Este ejemplo devuelve 873.720, 873.700, 873.000, 870.000, 800.000 y 0.000.

TYPE_ID



El esquema es SYSIBM.

La función TYPE_ID devuelve el identificador de tipo interno del tipo de datos dinámico de la *expresión*.

El argumento debe ser un tipo estructurado definido por el usuario. (Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos estructurado como argumento, no es necesario crear signaturas adicionales para dar soporte a los diferentes tipos definidos por el usuario).

El tipo de datos del resultado de la función es INTEGER. Si *expresión* puede tener un valor nulo, el resultado puede ser nulo; si *expresión* tiene un valor nulo, el resultado es el valor nulo.

El valor devuelto por la función TYPE_ID no es portátil a través de las bases de datos. El valor puede ser diferente aunque el esquema de tipo y el nombre de tipo del tipo de datos dinámico sean iguales. Cuando especifique el código para permitir la portabilidad, utilice las funciones TYPE_SCHEMA y TYPE_NAME para determinar el esquema de tipo y el nombre de tipo.

Ejemplos:

- Existe una jerarquía de tablas que tiene una tabla raíz EMPLOYEE de tipo EMP y una subtabla MANAGER de tipo MGR. Otra tabla ACTIVITIES incluye una columna denominada WHO_RESPONSIBLE que se define como REF(EMP) SCOPE EMPLOYEE. Para cada referencia de ACTIVITIES, visualice el identificador de tipo interno de la fila que corresponda a la referencia.

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
      TYPE_ID(DEREF(WHO_RESPONSIBLE))
    FROM ACTIVITIES
  ```

  La función DEREF se utiliza para devolver el objeto correspondiente a la fila.

TYPE_NAME

El esquema es SYSIBM.

La función TYPE_NAME devuelve el nombre no calificado del tipo de datos dinámico de la *expresión*.

El argumento debe ser un tipo estructurado definido por el usuario. (Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos estructurado como argumento, no es necesario crear signaturas adicionales para dar soporte a los diferentes tipos definidos por el usuario).

El tipo de datos del resultado de la función es VARCHAR(18). Si *expresión* puede tener un valor nulo, el resultado puede ser nulo; si *expresión* tiene un valor nulo, el resultado es el valor nulo. Utilice la función TYPE_SCHEMA para determinar el nombre de esquema del nombre de tipo devuelto por TYPE_NAME.

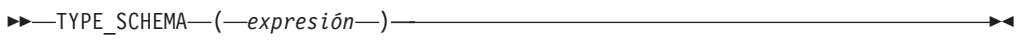
Ejemplos:

- Existe una jerarquía de tablas que tiene una tabla raíz EMPLOYEE de tipo EMP y una subtabla MANAGER de tipo MGR. Otra tabla ACTIVITIES incluye una columna denominada WHO_RESPONSIBLE que se define como REF(EMP) SCOPE EMPLOYEE. Para cada referencia de ACTIVITIES, visualice el tipo de la fila que corresponda a la referencia.

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
             TYPE_NAME(DEREF(WHO_RESPONSIBLE)),
             TYPE_SCHEMA(DEREF(WHO_RESPONSIBLE))
    FROM ACTIVITIES
  ```

  La función DEREF se utiliza para devolver el objeto correspondiente a la fila.

TYPE_SCHEMA



El esquema es SYSIBM.

La función TYPE_SCHEMA devuelve el nombre de esquema del tipo de datos dinámico de la *expresión*.

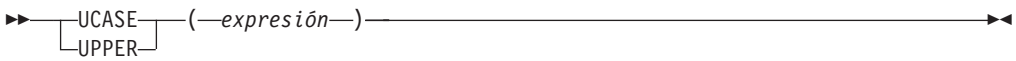
El argumento debe ser un tipo estructurado definido por el usuario. Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos estructurado como argumento, no es necesario crear signaturas adicionales para dar soporte a los diferentes tipos definidos por el usuario.

El tipo de datos del resultado de la función es VARCHAR(128). Si *expresión* puede tener un valor nulo, el resultado puede ser nulo; si *expresión* tiene un valor nulo, el resultado es el valor nulo. Utilice la función TYPE_NAME para determinar el nombre de tipo asociado con el nombre de esquema devuelto por TYPE_SCHEMA.

Información relacionada:

• "TYPE_NAME" en la página 452

UCASE o UPPER



El esquema es SYSIBM. (La versión SYSFUN de esta función sigue estando disponible para la compatibilidad con versiones superiores. Para obtener una descripción, consulte la documentación de la Versión 5.

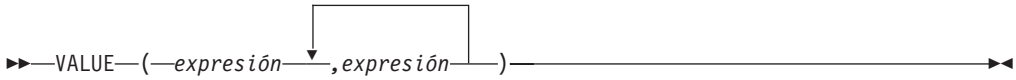
La función UCASE o UPPER es idéntica a la función TRANSLATE, excepto que sólo se especifica el primer argumento (*exp-serie-car*).

En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

Información relacionada:

• "TRANSLATE" en la página 448

VALUE



El esquema es SYSIBM.

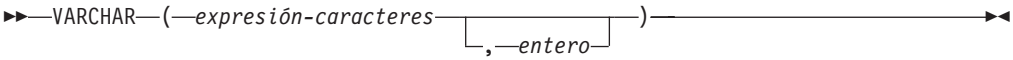
La función VALUE devuelve el primer argumento que no es nulo.

VALUE es sinónimo de COALESCE.

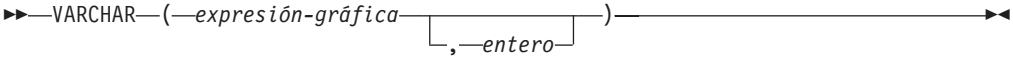
Información relacionada:
- "COALESCE" en la página 311

VARCHAR

De caracteres a varchar:



De gráfico a varchar:



De fecha y hora a varchar:

El esquema es SYSIBM.

La función VARCHAR devuelve una representación de serie de caracteres de longitud variable de:

- Una serie de caracteres, si el primer argumento es cualquier tipo de serie de caracteres
- Una serie gráfica (sólo para bases de datos Unicode), si el primer argumento es cualquier tipo de serie gráfica
- Un valor de fecha y hora, si el argumento es una fecha, una hora o una indicación de fecha y hora

En una base de datos Unicode, cuando en la serie de salida un carácter de varios bytes aparece truncado en parte:

- Si la salida era una serie de caracteres, el carácter parcial se sustituye por uno o varios blancos
- Si la salida era una serie gráfica, el carácter parcial se sustituye por una serie vacía

No confíe en ninguno de estos comportamientos, porque podrían cambiar en los releases futuros.

El resultado de la función es una serie de caracteres de longitud variable. Si el primer argumento puede ser nulo, el resultado puede ser nulo. Si el primer argumento es nulo, el resultado es el valor nulo.

De caracteres a varchar

expresión-caracteres
Una expresión cuyo valor debe ser de tipo de datos de serie de caracteres, con una longitud máxima de 32 672 bytes.

entero
El atributo de longitud de la serie de caracteres de longitud variable resultante. El valor debe estar entre 0 y 32 672. Si no se especifica este argumento, el atributo de longitud del resultado es igual al atributo de longitud del argumento.

De gráfico a varchar

expresión-gráfica
Una expresión cuyo valor debe ser de tipo de datos de serie gráfica distinto de LONG VARGRAPHIC o DBCLOB y cuya longitud máxima es de 16 336 caracteres de doble byte.

entero
El atributo de longitud de la serie de caracteres de longitud variable resultante. El valor debe estar entre 0 y 32 672. Si no se especifica este argumento, el atributo de longitud del resultado es igual al atributo de longitud del argumento.

De fecha y hora a varchar

expresión-fechahora
Una expresión cuyo valor debe ser del tipo de datos DATE, TIME o TIMESTAMP.

Ejemplo:

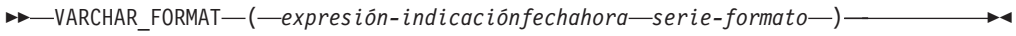
- Establezca la variable del lenguaje principal JOB_DESC, definida como VARCHAR(8), en el equivalente VARCHAR de la descripción del trabajo (que es el valor de la columna JOB), definido como CHAR(8), para la empleada Dolores Quintana.

```
SELECT VARCHAR(JOB)
  INTO :JOB_DESC
  FROM EMPLOYEE
  WHERE LASTNAME = 'QUINTANA'
```

Información relacionada:

- "CHAR" en la página 304

VARCHAR_FORMAT



El esquema es SYSIBM.

La función VARCHAR_FORMAT devuelve una representación de caracteres de una indicación de fecha y hora a la que se ha dado formato utilizando una plantilla de caracteres. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

expresión-indicaciónfechahora
Una expresión que da como resultado una indicación de fecha y hora. El argumento debe ser una indicación de fecha y hora o una representación de serie de una indicación de fecha y hora que no sea ni CLOB ni LONG VARCHAR. (Si *expresión-serie* es un marcador de parámetros sin tipo, se supone que el tipo es TIMESTAMP). La expresión de serie devuelve un valor CHAR o VARCHAR cuya longitud máxima no es mayor que 254 (SQLSTATE 42815). Los blancos iniciales y de cola se elimina de la *expresión-serie* y la subserie resultante se interpreta como una indicación de fecha y hora utilizando el formato especificado por la *serie-formato*. Los ceros iniciales pueden omitirse de todos los componentes de indicación de fecha y hora, con excepción del año. Es posible utilizar blancos en vez de los ceros iniciales para estos componentes. Por ejemplo, para una serie con el formato `'AAAA-MM-DD HH24:MI:SS'`, todas las series siguientes constituyen una especificación aceptable de las 9 de la mañana del 1 de enero de 2000:

```
'2000-1-01 09:00:00'    (un solo dígito para el mes)
'2000-1-01 09:00:00'    (un solo dígito - precedido por un blanco -
'2000-1-1 09:00:00'     (un solo dígito para el mes y el día)
'2000-01-01 9:00:00'    (un solo dígito para la hora)
'2000-01-01 09:0:0'     (un solo dígito para los minutos y los
                         segundos)
'2000- 1- 1 09: 0: 0'   (un solo dígito - precedido por un blanco -
                         para el mes, el día, los minutos y los
                         segundos)
'2000-01-01 09:00:00'   (número máximo de dígitos para cada elemento)
```

serie-formato
Una constante de caracteres que contiene una plantilla para el formato que deben darse al resultado. La longitud de la serie de formato no debe ser mayor que 254 (SQLSTATE 42815). Los blancos iniciales y de cola se elimina de la *expresión-formato* y la subserie resultante debe ser una plantilla válida para un valor de indicación de fecha y hora (SQLSTATE 42815). El contenido de *serie-formato* puede especificarse de forma mixta.

Las series de formato válidas son:

```
'AAAA-MM-DD HH24:MI:SS'
```

donde *AAAA* representa un valor de año de 4 dígitos; *MM* representa un valor de mes de 2 dígitos (01-12; enero=01); *DD* representa un valor de día del mes de 2 dígitos (01-31); *HH24* representa un valor de hora del día de 2 dígitos (00-24); Si la hora es 24, el valor de los minutos y de los segundos es cero.); *MI* representa un valor de minutos de 2 dígitos (00-59); y *SS* representa un valor de segundos 2 dígitos (00-59).

El resultado de la función es una serie de caracteres de longitud variable que contiene una expresión de indicación de fecha y hora formateada. La serie de formato también determina el atributo de longitud y la longitud real del resultado. Si la serie-formato es 'AAAA-MM-DD HH24:MI:SS', el atributo de longitud es 19. El resultado lo forman 19 caracteres de la forma siguiente:

```
AAAA-MM-DD HH:MI:SS
```

Por ejemplo, con el formato 'AAAA-MM-DD HH24:MI:SS' y una hora y fecha de las 10 de la mañana del 1 de enero de 2000, se devuelve lo siguiente:

```
'2000-01-01 10:00:00'
```

Aunque los valores para el mes y el día sólo requieren un solo dígito, en este ejemplo, cada dígito significativo va precedido de un cero inicial. Y, aunque el valor tanto para los minutos como para los segundos sea cero, se utiliza el número máximo de dígitos para cada uno y se devuelve '00' para cada una de estas partes en el resultado.

Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo. El CCSID del resultado es el CCSID de SBCS del sistema.

Ejemplo:

- Visualice los nombres de tabla y la indicación de fecha y hora de creación de todas las tablas del sistema cuyo nombres empiece por 'SYSU'.

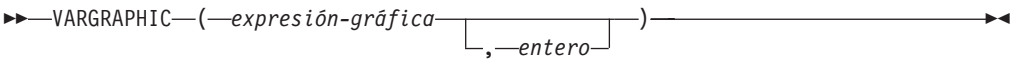
```
SELECT VARCHAR(name, 20) AS TABLE_NAME,
  VARCHAR_FORMAT(ctime, 'AAAA-MM-DD HH24:MI:SS') AS CREATION_TIME
   FROM SYSCAT.TABLES
  WHERE name LIKE 'SYSU%'
```

  Este ejemplo devuelve lo siguiente:

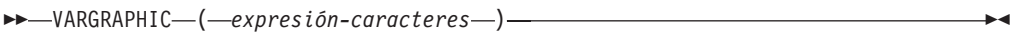
```
 TABLE_NAME           CREATION_TIME
-------------------- -------------------
SYSUSERAUTH          2000-05-19 08:18:56
SYSUSEROPTIONS       2000-05-19 08:18:56
```

VARGRAPHIC

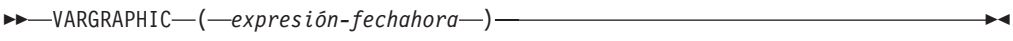
De gráfico a vargraphic:



De caracteres a vargraphic:



De fecha y hora a vargraphic:



El esquema es SYSIBM.

La función VARGRAPHIC devuelve una representación de serie gráfica de longitud variable de:

- Una serie gráfica, si el primer argumento es cualquier tipo de serie gráfica
- Una serie de caracteres, convirtiendo los caracteres de un solo byte en caracteres de doble byte, si el primer argumento es cualquier tipo de serie de caracteres
- Un valor de fecha y hora (sólo para bases de datos Unicode), si el argumento es una fecha, una hora o una indicación de fecha y hora

En una base de datos Unicode, si un argumento proporcionado es una serie de caracteres, se convertirá a una serie gráfica antes de que se ejecute la función. Cuando la serie de salida se trunca, de forma que el último carácter es un carácter de sustitución elevado, dicho carácter se convierte en un carácter en blanco (X'0020'). No confíe en este comportamiento, porque podría cambiar en los releases futuros.

El resultado de la función es una serie gráfica de longitud variable (tipo de datos VARGRAPHIC). Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

De gráfico a vargraphic

expresión-gráfica
Una expresión que devuelve un valor que es una serie gráfica.

entero
Un valor entero que especifica el atributo de longitud del tipo de datos VARGRAPHIC resultante. El valor debe estar entre 0 y 16 336. Si no se especifica ningún valor, el atributo de longitud del resultado es igual al atributo de longitud del primer argumento.

Si la longitud de la expresión gráfica es mayor que el atributo de longitud del resultado, el resultado se trunca. Se devuelve un aviso (SQLSTATE 01004) a menos que los caracteres truncados fuesen todos blancos y la expresión gráfica no fuese una serie larga (LONG VARGRAPHIC o DBCLOB).

De carácter a vargraphic

expresión-caracteres
Una expresión cuyo valor debe ser de tipo de datos de serie de caracteres distinto de LONG VARCHAR o CLOB y cuya longitud máxima es de 16 336 bytes.

El atributo de longitud del resultado es igual al atributo de longitud del argumento.

En el resultado, todos los caracteres de un solo byte de la *expresión-caracteres* se convierten en su representación de doble byte equivalente o en el carácter de sustitución de doble byte. Todos los caracteres de doble byte de la *expresión-caracteres* se correlacionan sin ninguna conversión adicional. Si el primer byte de un carácter de doble byte aparece como el último byte de la *expresión-caracteres*, éste se convierte en el carácter de sustitución de doble byte. El orden secuencial de los caracteres de la *expresión-caracteres* se conserva.

En una base de datos Unicode, esta función convierte la serie de caracteres de la página de códigos del operando a UCS-2. Se convierte cada carácter del operando, incluidos los caracteres de doble byte. Si se proporciona un valor para el segundo argumento, éste especifica la longitud necesaria de la serie resultante (en caracteres UCS-2).

La conversión a elementos de código de doble byte por la función VARGRAPHIC se basa en la página de códigos del operando.

Los caracteres de doble byte del operando no se convierten. El resto de caracteres se convierten a sus equivalentes de doble byte correspondientes. Si no existe un equivalente de doble byte correspondiente, se utiliza el carácter de sustitución de doble byte para la página de códigos.

No se genera ningún aviso ni código de error si se devuelven uno o varios caracteres de sustitución de doble byte en el resultado.

De fecha y hora a vargraphic

expresión-fechahora
Una expresión cuyo valor debe ser del tipo de datos DATE, TIME o TIMESTAMP.

Información relacionada:

- "GRAPHIC" en la página 366
- Apéndice N, "Consideraciones sobre el código UNIX ampliado (EUC) en japonés y chino tradicional", en la página 801

WEEK

Devuelve la semana del año del argumento como un valor entero en el rango de 1 a 54. La semana empieza en domingo.

El argumento debe ser una fecha, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o de una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

WEEK_ISO

El esquema es SYSFUN.

Devuelve la semana del año del argumento como un valor entero en el rango de 1 a 53. La semana empieza en lunes e incluye siempre 7 días. La semana 1 es la primera semana del año que contenga un jueves, que equivale a la primera semana que contenga el 4 de enero. Por consiguiente, es posible hacer que aparezca un máximo de 3 días del principio de un año en la última semana del año anterior. Y, a la inversa, pueden aparecer un máximo de 3 días del final de un año en la primera semana del año siguiente.

El argumento debe ser una fecha, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o de una indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

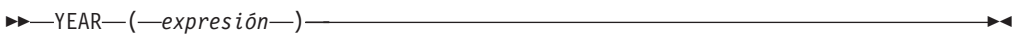
El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

La lista siguiente muestra ejemplos del resultado de WEEK_ISO y DAYOFWEEK_ISO.

```
DATE       WEEK_ISO    DAYOFWEEK_ISO
---------- ----------- -------------
1997-12-28          52             7
1997-12-31           1             3
1998-01-01           1             4
1999-01-01          53             5
1999-01-04           1             1
1999-12-31          52             5
2000-01-01          52             6
2000-01-03           1             1
```

YEAR



El esquema es SYSIBM.

La función YEAR devuelve la parte correspondiente al año de un valor.

El argumento debe ser una fecha, una indicación de fecha y hora, una duración de fecha, una duración de indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o indicación de fecha y hora que no sea CLOB ni LONG VARCHAR. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento especificado:

- Si el argumento es una fecha, una indicación de fecha y hora o una representación válida de una fecha o de una fecha y hora en forma de serie:
  - El resultado es la parte correspondiente al año del valor, que es un entero entre 1 y 9.999.
- Si el argumento es una duración de fecha o duración de indicación de fecha y hora:
  - El resultado es la parte correspondiente al año del valor, que es un entero entre –9.999 y 9.999. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplos:

- Seleccione todos los proyectos de la tabla PROJECT que se han planificado para empezar (PRSTDATE) y finalizar (PRENDATE) en el mismo año.

```
SELECT * FROM PROJECT
  WHERE YEAR(PRSTDATE) = YEAR(PRENDATE)
```

- Seleccione todos los proyectos de la tabla PROJECT que se haya planificado que finalizasen en menos de un año.

```
SELECT * FROM PROJECT
  WHERE YEAR(PRENDATE - PRSTDATE) < 1
```

Funciones de tabla

Sólo se puede utilizar una función de tabla en la cláusula FROM de una sentencia. Las funciones de tabla devuelven columnas de una tabla, de aspecto parecido a una tabla creada mediante una sentencia CREATE TABLE simple. Las funciones de tabla pueden calificarse con un nombre de esquema.

Todas las funciones de tabla de supervisión utilizan una conexión de instancias independiente, distinta de la que utiliza la sesión actual. Por lo tanto, sólo entran en vigor los conmutadores de supervisor del gestor de bases de datos por omisión. Los conmutadores de supervisor activados o desactivados dinámicamente de la sesión o aplicación actual no entran en vigor.

Procedimientos

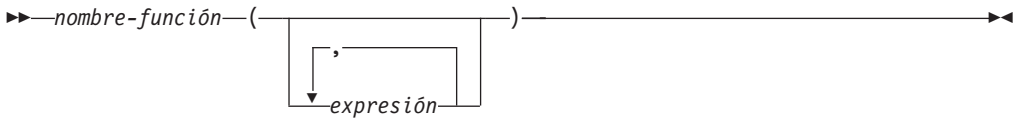
Un procedimiento es un programa de aplicación que se puede iniciar mediante la sentencia CALL de SQL. El procedimiento se especifica mediante un nombre de procedimiento, que puede ir seguido por argumentos, incluidos entre paréntesis.

El argumento o los argumentos de un procedimiento son valores escalares individuales, que pueden ser de tipos diferentes y pueden tener significados diferentes. Los argumentos pueden utilizarse para pasar valores en el procedimiento y recibir y/o devolver valores del procedimiento.

Los procedimientos definidos por el usuario son procedimientos que se registran en una base de datos en SYSCAT.ROUTINES, utilizando la sentencia CREATE PROCEDURE. Con el gestor de bases de datos se proporciona un conjunto de funciones de este tipo, en un esquema llamado SYSFUN y otro en un esquema llamado SYSPROC.

Los procedimientos pueden calificarse con el nombre de esquema.

Funciones definidas por el usuario



Las *funciones definidas por el usuario (UDF)* son extensiones o adiciones a las funciones incorporadas existentes del lenguaje SQL. Una función definida por el usuario puede ser una función escalar, que devuelve un solo valor cada vez que se invoca; una función de columna, a la que se pasa un conjunto de valores similares y devuelve un solo valor para el conjunto; una función de fila, que devuelve una fila o una función de tabla, que devuelve una tabla.

En los esquemas SYSFUN y SYSPROC se proporcionan varias funciones definidas por el usuario.

Una UDF puede ser una función de columna sólo si se deriva de una función de columna existente. Se hace referencia a una UDF mediante un nombre de función calificado o no calificado, seguido de paréntesis que encierran los argumentos de la función (si los hay). Una función de columna escalar definida por el usuario registrada con la base de datos puede aludirse en los mismos contextos en que pueda aparecer cualquier función incorporada. Una función de fila definida por el usuario sólo puede aludirse implícitamente cuando está registrada como función de transformación para un tipo definido por el usuario. Una función de tabla definida por el usuario registrada en la base de datos sólo puede aludirse en la cláusula FROM de una sentencia SELECT.

Los argumentos de la función deben corresponderse en número y posición con los parámetros especificados para la función definida por el usuario cuando se registró con la base de datos. Además, los argumentos deben ser de tipos de datos que sean promocionables a los tipos de datos de los parámetros definidos correspondientes.

El resultado de la función es el especificado en la cláusula RETURNS. La cláusula RETURNS, definida cuando se registró la UDF, determina si una función es una función de tabla o no lo es. Si se especifica (o se toma como valor por omisión) la cláusula RETURNS NULL ON NULL INPUT al registrar la función, el resultado es nulo si algún argumento es nulo. En el caso de las funciones de tabla, esto se interpreta como una tabla de retorno sin filas (es decir, una tabla vacía).

A continuación se muestran algunos ejemplos de funciones definidas por el usuario:

- Una UDF escalar denominada ADDRESS extrae la dirección de inicio de los resúmenes almacenados en formato script. La función ADDRESS espera un argumento CLOB y devuelve VARCHAR(4000):

  ```
  SELECT EMPNO, ADDRESS(RESUME) FROM EMP_RESUME
    WHERE RESUME_FORMAT = 'SCRIPT'
  ```

- La tabla T2 tiene una columna numérica A. Al invocar la UDF escalar denominada ADDRESS del ejemplo anterior:

  ```
  SELECT ADDRESS(A) FROM T2
  ```

  se genera un error (SQLSTATE 42884), ya que no existe ninguna función con un nombre que coincida y con un parámetro promocionable del argumento.

- Una UDF de tabla denominada WHO devuelve información acerca de las sesiones de la máquina servidora que estaban activas en el momento de ejecutar la sentencia. La función WHO se invoca desde una cláusula FROM que incluye la palabra clave TABLE y una variable de correlación obligatoria. Los nombres de columna de la tabla WHO() se han definido en la sentencia CREATE FUNCTION.

```
SELECT ID, START_DATE, ORIG_MACHINE
FROM TABLE( WHO() ) AS QQ
WHERE START_DATE LIKE 'MAY%'
```

Información relacionada:

- "Subselección" en la página 470
- "Sentencia CREATE FUNCTION" en la publicación *Consulta de SQL, Volumen 2*

Capítulo 4. Consultas

Consultas de SQL

Una *consulta* especifica una tabla resultante. Una consulta es un componente de algunas sentencias de SQL. Las tres formas de una consulta son:

- subselección
- selección completa
- sentencia-select.

Autorización

El ID de autorización de la sentencia debe tener como mínimo uno de los siguientes privilegios o autorizaciones para cada tabla, vista o apodo al que la consulta haga referencia:

- Autorización SYSADM o DBADM
- Privilegio CONTROL
- Privilegio SELECT.

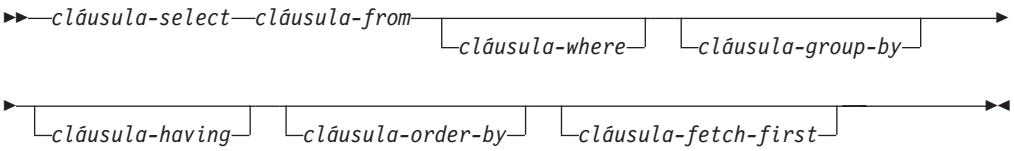
Los privilegios de grupo, con excepción de PUBLIC, no se comprueban para las consultas contenidas en sentencias de SQL estático.

Para los apodos, los requisitos de autorización de la fuente de datos para el objeto al que el apodo hace referencia se aplican cuando se procesa la consulta. El ID de autorización de la sentencia puede estar correlacionado con un ID de autorización diferente en la fuente de datos.

Información relacionada:

- "Sentencia SELECT INTO" en la publicación *Consulta de SQL, Volumen 2*

Subselección



La *subselección* es un componente de la selección completa.

Una subselección especifica una tabla resultante que deriva de las tablas, vistas o apodos identificados en la cláusula FROM. La derivación puede describirse como una secuencia de operaciones en las que el resultado de cada operación es la entrada de la siguiente. (Es la única manera de describir la subselección. El método utilizado para realizar la derivación puede ser bastante diferente de esta descripción.)

Las cláusulas de la subselección se procesan en el orden siguiente:

1. cláusula FROM
2. cláusula WHERE
3. cláusula GROUP BY
4. cláusula HAVING
5. cláusula SELECT
6. cláusula ORDER BY
7. Cláusula FETCH FIRST

Una subselección que contenga una cláusula ORDER BY o FETCH FIRST no puede especificarse:

- En la selección completa más exterior de una vista.
- En una tabla de consulta materializada.
- A menos que la subselección esté entre paréntesis.

Por ejemplo, lo siguiente no es válido (SQLSTATE 428FJ):

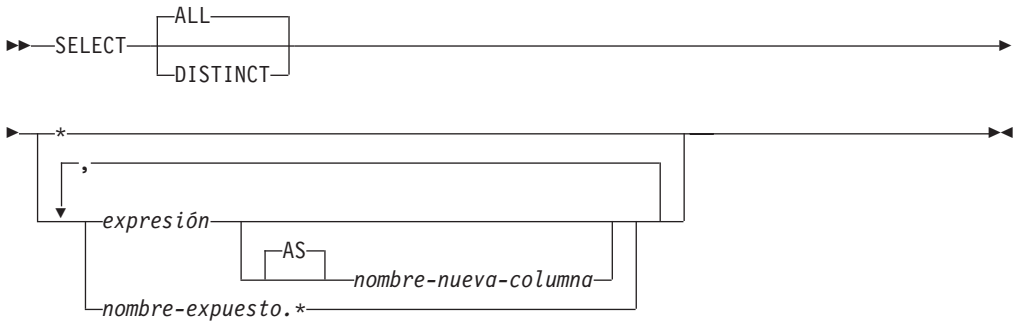
```
SELECT * FROM T1
   ORDER BY C1
UNION
SELECT * FROM T2
   ORDER BY C1
```

El ejemplo siguiente *sí* es válido:

```
(SELECT * FROM T1
   ORDER BY C1)
UNION
(SELECT * FROM T2
   ORDER BY C1)
```

Nota: Una cláusula ORDER BY en una subselección no afecta el orden de las filas que una consulta devuelve. Una cláusula ORDER BY sólo afecta el orden de las filas devueltas si se especifica en la selección completa más externa.

cláusula-select



La cláusula SELECT especifica las columnas de la tabla resultante final. Los valores de columna los genera la aplicación de la *lista de selección* a R. La lista de selección son los nombres o expresiones especificados en la cláusula SELECT y R es el resultado de la operación anterior de la subselección. Por ejemplo, si las únicas cláusulas especificadas con SELECT, FROM y WHERE, R es el resultado de la cláusula WHERE.

ALL
Retiene todas las filas de la tabla resultante final y no elimina los duplicados redundantes. Este es el valor por omisión.

DISTINCT
Elimina todas las filas excepto una de los juegos de filas duplicadas de la tabla resultante final. Si se utiliza DISTINCT, ninguna columna de tipo serie de la tabla resultante puede ser un tipo LONG VARCHAR, LONG VARGRAPHIC, DATALINK o LOB, un tipo diferenciado de cualquiera de estos tipos ni un tipo estructurado. DISTINCT puede utilizarse más de una vez en una subselección. Esto incluye SELECT DISTINCT, la utilización de DISTINCT en una función de columna de la lista de selección o la cláusula HAVING y las subconsultas de la subselección.

Dos filas sólo son duplicadas una de la otra si cada valor de la primera es igual al valor correspondiente de la segunda. Para la determinación de duplicados, dos valores nulos se consideran iguales.

Notación de lista de selección:

* Representa una lista de nombres que identifican las columnas de la tabla R. El primer nombre de la lista identifica la primera columna de R, el segundo nombre identifica la segunda columna de R y así sucesivamente.

La lista de nombres se establece cuando se vincula el programa que contiene la cláusula SELECT. Por lo tanto, * (el asterisco) no identifica ninguna columna que se haya añadido a la tabla después de vincular la sentencia que contiene la referencia a la tabla.

expresión
Especifica los valores de una columna del resultado. Puede ser cualquier expresión que sea un elemento válido en el lenguaje SQL pero normalmente incluye nombres de columnas. Cada nombre de columna utilizado en la lista de selección debe identificar sin ambigüedades una columna R.

nuevo-nombre-columna o **AS** *nuevo-nombre-columna*
Nombra o cambia el nombre de la columna del resultado. El nombre no

debe estar calificado y no tiene que ser exclusivo. El uso subsiguiente del nombre-columna está limitado en lo siguiente:

- Un nuevo-nombre-columna especificado en la cláusula AS se puede utilizar en la cláusula-order-by, siempre que sea exclusivo.
- Un nuevo-nombre-columna especificado en la cláusula AS de la lista de selección no se puede utilizar en ninguna otra cláusula de la subselección (cláusula-where, cláusula-group-by o cláusula-having).
- Un nuevo-nombre-columna especificado en la cláusula AS no se puede utilizar en la cláusula-update.
- Un nuevo-nombre-columna especificado en la cláusula AS se conoce fuera de la selección completa de las expresiones de tabla anidadas, las expresiones de tablas comunes y CREATE VIEW.

*nombre.**

Representa la lista de nombres que identifican las columnas de la tabla resultante identificada por *nombre-expuesto*. El *nombre-expuesto* puede ser un nombre de tabla, un nombre de vista, un apodo o un nombre de correlación, y debe designar una tabla, una vista o un apodo especificado en la cláusula FROM. El primer nombre de la lista identifica la primera columna de la tabla, vista o apodo, el segundo nombre de la lista identifica la segunda columna de la tabla, vista o apodo, y así sucesivamente.

La lista de nombres se establece cuando se vincula la sentencia que contiene la cláusula SELECT. Por lo tanto, * no identifica ninguna columna que se haya añadido a la tabla después de vincular la sentencia.

El número de columnas del resultado de SELECT es igual al número de expresiones de la forma operativa de la lista de selección (es decir, la lista establecida cuando se ha preparado la sentencia) y no puede exceder de 500 para una página de 4K de tamaño o de 1012 para una página de 8K, 16K o 32K de tamaño.

Límites en las columnas de una serie

Para ver las limitaciones en la lista de selección, consulte “Restricciones al utilizar series de caracteres de longitud variable”.

Aplicación de la lista de selección

Algunos resultados de aplicar la lista de selección en R dependen de si se utiliza GROUP BY o HAVING o no. Los resultados se describen en dos listas separadas:

Si se utiliza GROUP BY o HAVING:

- Una expresión *X* (no una función de columna) utilizada en la lista de selección debe contener una cláusula GROUP BY con:
  - una *expresión-agrupación* en la cual cada nombre-columna identifique sin ambigüedades una columna de R (consulte el apartado “Cláusula group-by” en la página 484), o bien,
  - cada columna de R a la que *X* haga referencia como una *expresión-agrupación* separada.
- La lista de selección se aplica a cada grupo de R y el resultado contiene tantas filas como grupos haya en R. Cuando la lista de selección se aplica a un grupo de R, este grupo es la fuente de los argumentos de las funciones de columna de la lista de selección.

Si no se utilizan ni GROUP BY ni HAVING:

- La lista de selección no debe incluir ninguna función de columna o cada *nombre-columna* de la lista de selección debe estar especificado en una función de columna o debe ser una referencia de columna correlacionada.
- Si la selección no incluye funciones de columna, la lista de selección se aplica a cada fila de R y el resultado contiene tantas filas como el número de filas en R.
- Si la lista de selección es una lista de funciones de columna, R es la fuente de los argumentos de las funciones y el resultado de aplicar la lista de selección es una fila.

En cualquier caso la columna *n* del resultado contiene los valores especificados por la aplicación de la expresión *n* en la forma operativa de la lista de selección.

Atributos nulos de las columnas del resultado: Las columnas del resultado no permiten valores nulos si se derivan de:

- Una columna que no permite valores nulos
- Una constante
- La función COUNT o COUNT_BIG
- Una variable del lenguaje principal que no tiene una variable indicadora
- Una función o expresión escalar que no incluye un operando que permita nulos.

Las columnas del resultado permiten valores nulos si se derivan de:

- Cualquier función de columna excepto COUNT o COUNT_BIG
- Una columna que permite valores nulos
- Una función o expresión escalar que incluye un operando que permite nulos
- Una función NULLIF con argumentos que contienen valores iguales.
- Una variable del lenguaje principal que contiene una variable indicadora.
- Un resultado de una operación de conjuntos si como mínimo uno de los elementos correspondientes de la lista de selección admite nulos.
- Una expresión aritmética o columna de vista que se deriva de una expresión aritmética y la base de datos está configurada con DFT_SQLMATHWARN establecido en ″yes″
- Una operación de desreferencia.

Nombres de las columnas del resultado:

- Si se especifica la cláusula AS, el nombre de la columna del resultado es el nombre especificado en esta cláusula.
- Si no se especifica la cláusula AS y la columna del resultado se deriva de una columna, el nombre de columna del resultado es el nombre no calificado de dicha columna.
- Si no se especifica la cláusula AS y la columna del resultado se deriva mediante una operación de desreferencia, el nombre de columna del resultado es el nombre no calificado de la columna de destino de la operación de desreferencia.
- Todos los demás nombres de columnas del resultado carecen de nombre. El sistema asigna números temporales (como series de caracteres) a estas columnas.

Tipos de datos de las columnas del resultado: Cada columna del resultado de SELECT adquiere un tipo de datos de la expresión de la que se deriva.

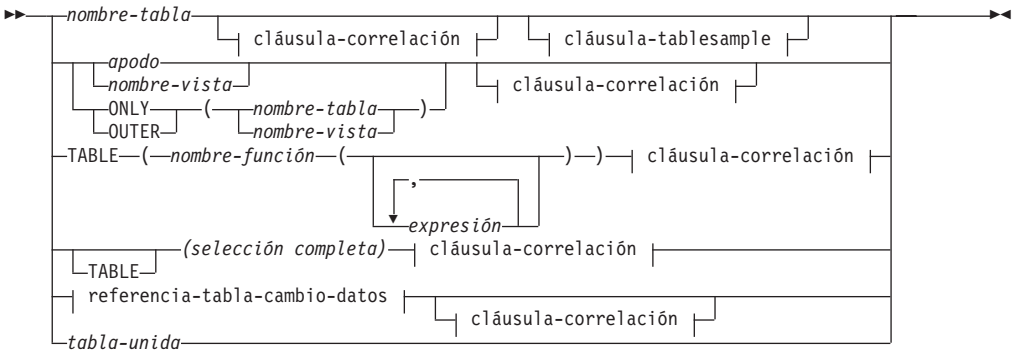
Cuando la expresión es ...	El tipo de datos de la columna del resultado es ...
el nombre de cualquier columna numérica	el mismo que el tipo de datos de la columna, con la misma precisión y escala que para las columnas DECIMAL.
una constante de enteros	INTEGER.
una constante decimal	DECIMAL, con la precisión y la escala de la constante.
una constante de coma flotante	DOUBLE.
el nombre de cualquier variable numérica	el mismo que el tipo de datos de la variable, con la misma precisión y escala que para las variables DECIMAL.
una constante hexadecimal que representa n bytes	VARCHAR(n); la página de códigos es la página de códigos de la base de datos.
el nombre de cualquier columna de serie	el mismo que el tipo de datos de la columna, con el mismo atributo de longitud.
el nombre de cualquier variable de serie.	el mismo que el tipo de datos de la variable, con el mismo atributo de longitud; si el tipo de datos de la variable no es idéntico a un tipo de datos SQL (por ejemplo, una serie terminada en NUL en C), la columna del resultado es una serie de longitud variable.
una constante de serie de caracteres de longitud *n*	VARCHAR(*n*).
una constante de serie gráfica de longitud *n*	VARGRAPHIC(*n*).
el nombre de una columna de indicación de fecha y hora	el mismo tipo de datos de la columna.
el nombre de una columna de tipo definido por el usuario	el mismo tipo de datos de la columna.
el nombre de una columna de tipo de referencia	el mismo tipo de datos de la columna.

cláusula-from

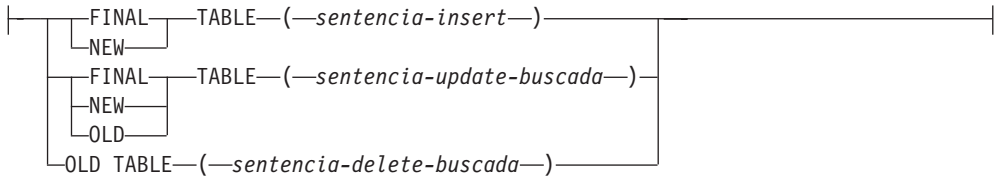
La cláusula FROM especifica una tabla resultante intermedia.

Si se especifica una referencia-tabla, la tabla resultante intermedia es simplemente el resultado de dicha referencia-tabla. Si se especifica más de una referencia-tabla, la tabla resultante intermedia consta de todas las combinaciones posibles de las filas de las referencias-tabla especificadas (el producto cartesiano). Cada fila del resultado es una fila de la primera referencia-tabla concatenada con una fila de la segunda referencia-tabla concatenada a su vez con una fila de la tercera, etcétera. El número de filas del resultado es el producto del número de filas de todas las referencias-tabla individuales. Para obtener una descripción de *referencia-tabla*, consulte “referencia-tabla” en la página 475.

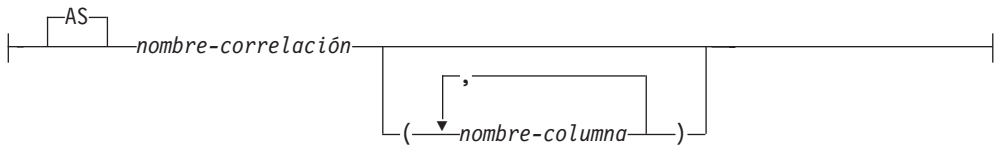
referencia-tabla



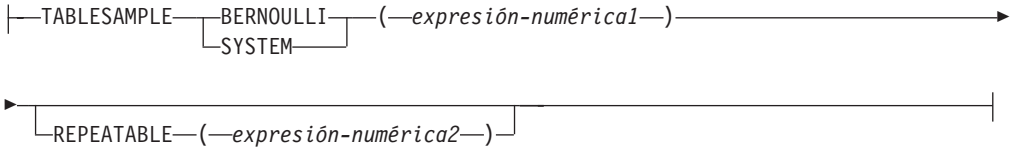
referencia-tabla-cambio-datos:



cláusula-correlación:



cláusula-tablesample:



Cada *nombre-tabla*, *nombre-vista* o *apodo* especificado como referencia-tabla debe identificar una tabla, vista o apodo existente del servidor de aplicaciones o el *nombre-tabla* de una expresión de tabla común definida antes de la selección completa que contiene la referencia-tabla. Si *nombre-tabla* hace referencia a una tabla con tipo, el nombre indica UNION ALL de la tabla con todas sus subtablas y solamente con las columnas de *nombre-tabla*. De manera similar, si *nombre-vista* hace referencia a una vista con tipo, el nombre indica UNION ALL de la vista con todas sus subvistas y solamente con las columnas de *nombre-vista*.

El uso de ONLY(*nombre-tabla*) u ONLY(*nombre-vista*) significa que no se incluyen las filas de las subtablas o subvistas correspondientes. Si el *nombre-tabla* utilizado con ONLY no tiene subtablas, ONLY(*nombre-tabla*) equivale a especificar *nombre-tabla*. Si el *nombre-vista* utilizado con ONLY no tiene subvistas, ONLY(*nombre-vista*) equivale a especificar *nombre-vista*.

El uso de OUTER(*nombre-tabla*) u OUTER(*nombre-vista*) representa una tabla virtual. Si el *nombre-tabla* o el *nombre-vista* que se utilice con OUTER no tiene subtablas o subvistas, especificar OUTER equivale a no especificar OUTER.
OUTER(*nombre-tabla*) se deriva de *nombre-tabla* de la manera siguiente:

- En las columnas, se incluyen las columnas de *nombre-tabla* seguidas de las columnas adicionales que ha introducido cada una de sus subtablas (si existen). Las columnas adicionales se añaden a la derecha atravesando la jerarquía de las subtablas por orden de importancia. Se atraviesan las subtablas que tienen un padre común en el orden de creación de sus tipos.
- En las filas, se incluyen todas las filas de *nombre-tabla* y todas las filas de sus subtablas. Se devuelven valores nulos para las columnas que no están en la subtabla para la fila.

Los puntos anteriores también se aplican a OUTER(*nombre-vista*) si se sustituye *nombre-tabla* por *nombre-vista* y subtabla por subvista.

El uso de ONLY o OUTER requiere el privilegio SELECT en cada subtabla de *nombre-tabla* o subvista de *nombre-vista*.

Cada *nombre-función*, junto con los tipos de sus argumentos, especificado como una referencia a tabla, debe resolverse en una función de tabla existente en el servidor de aplicaciones.

Una selección completa entre paréntesis seguida de un nombre de correlación se llama una *expresión de tabla anidada*.

Una *tabla-unida* especifica un conjunto resultante intermedio que es el resultado de una o varias operaciones de unión. Para obtener información, vea “tabla-unida” en la página 482.

Los nombres expuestos de todas las referencias a tabla deben ser exclusivos. Un nombre expuesto es:

- Un *nombre-correlación*,
- Un *nombre-tabla* que no va seguido de un *nombre-correlación*,
- Un *nombre-vista* que no va seguido de un *nombre-correlación*,
- Un *apodo* que no va seguido de un *nombre-correlación*,
- Un *nombre-seudónimo* que no va seguido de un *nombre-correlación*.

Cada *nombre-correlación* se define como un designador de la referencia inmediatamente precedente de *nombre-tabla*, *nombre-vista*, *apodo*, *nombre-función* o expresión de tabla anidada. Una referencia calificada a una columna para una tabla, vista, función de tabla o expresión de tabla anidada debe utilizar el nombre expuesto. Si se especifica dos veces el mismo nombre de tabla, vista o apodo, como mínimo una especificación debe ir seguida de un *nombre-correlación*. El *nombre-correlación* se utiliza para calificar las referencias a las columnas de la tabla, vista o apodo. Cuando se especifica un *nombre-correlación*, también se pueden especificar *nombres-columna* para asignar nombres a las columnas de la referencia a *nombre-tabla*, *nombre-vista*, *apodo*, *nombre-función* o expresión de tabla anidada.

En general, las funciones de tabla y las expresiones de tabla anidadas pueden especificarse en cualquier cláusula-from. Las columnas de las funciones de tabla y las expresiones de tabla anidadas pueden hacerse referencia en la lista de selección y en el resto de la subselección utilizando el nombre de correlación que debe especificarse. El ámbito de este nombre de correlación es el mismo que los

nombres de correlación para otros nombres de tabla, vista o apodo de la cláusula FROM. Una expresión de tabla anidada puede utilizarse:

- En el lugar de una vista para evitar la creación de la vista (cuando no es necesario el uso general de la vista)
- Cuando la tabla resultante deseada se basa en variables del lenguaje principal

Una expresión de la lista de selección de una expresión de tabla anidada a la que se hace referencia dentro, o que representa el destino de, una sentencia de cambio de datos dentro de una selección completa sólo es válida cuando no incluye:

- Una función que lee o modifica datos de SQL
- Una función que no es determinante
- Una función que tiene acción externa
- Una función OLAP

Si se hace referencia directamente a una vista en, o es el destino de, una expresión de tabla anidada de una sentencia de cambio de datos dentro de una cláusula FROM, la vista debe ser simétrica (debe tener especificado WITH CHECK OPTION) o debe satisfacer la restricción correspondiente a una vista WITH CHECK OPTION.

Si el destino de una sentencia de cambio de datos dentro de una cláusula FROM es una expresión de tabla anidada, las filas modificadas no se vuelven a calificar, los predicados de la cláusula WHERE no se vuelven a evaluar y las operaciones ORDER BY o FETCH FIRST no se vuelven a realizar.

La cláusula-tablesample opcional se puede utilizar para obtener un subconjunto aleatorio (un ejemplo) de las filas a partir del *nombre-tabla* especificado, en lugar del contenido completo de dicho *nombre-tabla*, para esta consulta. Este muestreo se añade a cualquier predicado especificado en la *cláusula-where*. A no ser que se especifique la cláusula REPEATABLE opcional, cada ejecución de la consulta generará un ejemplo distinto, excepto en casos en los que la tabla sea tan pequeña en relación al tamaño del ejemplo que cualquier ejemplo deba devolver las mismas filas. El tamaño del ejemplo se controla mediante *expresión-numérica1* entre paréntesis, que representa un porcentaje aproximado (P) de la tabla que se va a devolver. El método por el que se obtiene el ejemplo se especifica tras la palabra clave TABLESAMPLE y puede ser BERNOULLI o SYSTEM. Para ambos métodos, el número exacto de filas del ejemplo puede ser distinto para cada ejecución de la consulta, pero en promedio debe ser aproximadamente P por ciento de la tabla, antes de que los predicados reduzcan el número de filas.

El *nombre-tabla* debe ser una tabla almacenada. Puede ser un nombre de tabla de consulta materializada (MQT), pero no una subselección o expresión de tabla para la que se haya definido una MQT, porque no hay ninguna garantía de que el gestor de bases de datos vaya a direccionar a la MQT para dicha subselección.

Semánticamente, el muestreo de una tabla se produce antes de cualquier otro proceso de consulta, como por ejemplo aplicar predicados o realizar uniones. Los accesos repetidos de una tabla de muestreo dentro de una sola ejecución de una consulta (como en una unión de bucle anidado o una subconsulta correlacionada) devolverán el mismo ejemplo. Se pueden obtener ejemplos de más de una tabla en una consulta.

El muestreo BERNOULLI considera cada fila de forma individual. Incluye cada fila en el ejemplo con la probabilidad P/100 (donde P es el valor de

expresión-numérica1) y ejecuta cada fila con la probabilidad 1 - P/100, independientemente de las demás filas. De modo que si *expresión-numérica1* tiene el valor 10, lo que significa un ejemplo del diez por ciento, cada fila se incluiría con la probabilidad 0,1 y se excluiría con la probabilidad 0,9.

El muestreo SYSTEM permite al gestor de bases de datos determinar la forma más eficiente de realizar el muestreo. En la mayoría de los casos, el muestreo SYSTEM aplicado a *nombre-tabla* significa que cada página de *nombre-tabla* se incluye en el ejemplo con una probabilidad de P/100 y se excluye con una probabilidad de 1 - P/100. Todas las filas de cada página que se incluye están cualificadas para el ejemplo. El muestreo SYSTEM de *nombre-tabla* generalmente se ejecuta con mayor rapidez que el muestreo BERNOULLI porque se tienen que recuperar menos páginas de datos; sin embargo, a menudo puede dar lugar a estimaciones menos precisas para las funciones de agregación (SUM(SALES), por ejemplo), especialmente si las filas de *nombre-tabla* están en clúster en cualquier columna a la que se haga referencia en dicha consulta. En determinadas circunstancias, el optimizador puede decidir que es más eficiente realizar un muestreo SYSTEM como si fuera un muestreo BERNOULLI, por ejemplo cuando un índice puede aplicar un predicado en *nombre-tabla* y es mucho más selectivo que la tasa de muestreo P.

expresión-numérica1 especifica el tamaño del ejemplo que se obtendrá de *nombre-tabla*, expresado como un porcentaje. Debe ser una expresión numérica constante que no puede contener columnas, marcadores de parámetro ni variables del lenguaje principal. La expresión se debe evaluar en un número positivo menor o igual que 100, pero puede estar entre 1 y 0. Por ejemplo, el valor 0,01 representa una centésima de un porcentaje, lo que significa que se tomará un muestreo de 1 fila entre 10.000 como promedio. Una *expresión-numérica1* que se evalúe en 100 se maneja como si no se hubiera especificado la cláusula-tablesample. Si *expresión-numérica1* se evalúa en un valor nulo o en un valor mayor que 100 o menor que 0, se devuelve un error (SQLSTATE 2202H).

A veces resulta recomendable que el muestreo se repita de una ejecución de la consulta a la siguiente; por ejemplo, durante una prueba de regresión o "depuración" de la consulta. Esto se puede conseguir especificando la cláusula REPEATABLE. La cláusula REPEATABLE necesita la especificación de una *expresión-numérica2* entre paréntesis, que desempeña la misma función que el valor raíz de un generador de números aleatorios. La adición de la cláusula REPEATABLE a la cláusula-tablesample de cualquier *nombre-tabla* asegura que las ejecuciones repetidas de dicha consulta (utilizando el mismo valor para *expresión-numérica2*) devuelven el mismo ejemplo, dando por supuesto que los datos no se han actualizado, reorganizado ni reparticionado. Para garantizar que se utiliza el mismo ejemplo de *nombre-tabla* entre diversas consultas, se recomienza utilizar una tabla temporal global. Como alternativa, se pueden combinar varias consultas en una consulta, con varias referencias a un ejemplo definido mediante la cláusula WITH.

A continuación se muestran algunos ejemplos:

Ejemplo 1: Solicitar un ejemplo Bernoulli del 10% de la tabla Sales por motivos de auditoría.

```
SELECT * FROM Sales
  TABLESAMPLE BERNOULLI(10)
```

Ejemplo 2: Calculas los ingresos totales por ventas de la región Northeast para cada categoría de producto, utilizando un ejemplo aleatorio SYSTEM del 1% de la tabla

Sales. La semántica de SUM corresponde a la propia muestra, de modo que para extrapolar las ventas a la tabla Sales completa, la consulta debe dividir ducha SUM por la tasa de muestreo (0,01).

```
SELECT SUM(Sales.Revenue) / (0.01)
  FROM Sales TABLESAMPLE SYSTEM(1)
  WHERE Sales.RegionName = 'Northeast'
  GROUP BY Sales.ProductCategory
```

Ejemplo 3: Utilizando la cláusula REPEATABLE, modificar la consulta anterior para asegurar que se obtiene el mismo resultado (aleatorio) cada vez que se ejecuta la consulta. (El valor de la constante especificada entre paréntesis es arbitrario.)

```
SELECT SUM(Sales.Revenue) / (0.01)
  FROM Sales TABLESAMPLE SYSTEM(1) REPEATABLE(3578231)
  WHERE Sales.RegionName = 'Northeast'
  GROUP BY Sales.ProductCategory
```

Referencias a las funciones de tabla

En general, se puede hacer referencia a una función de tabla, junto a los valores de sus argumentos en la cláusula FROM de una sentencia SELECT, exactamente de la misma manera que una tabla o una vista. Sin embargo, se aplican algunas consideraciones especiales.

- Nombres de columna de función de tabla

  A menos que se proporcionen nombres de columna alternativos a continuación del *nombre-correlación*, los nombres de columna de la función de tabla son los especificados en la cláusula RETURNS de la sentencia CREATE FUNCTION. Es análogo a los nombres de las columnas de una tabla, que se definen en la sentencia CREATE TABLE.

- Resolución de una función de tabla

  Los argumentos especificados en una referencia de función de tabla, junto con el nombre de la función, se utilizan por un algoritmo llamado *resolución de función* para determinar la función exacta que se va a utilizar. Esta operación no es diferente de lo que ocurre con las demás funciones (por ejemplo, en las funciones escalares) utilizadas en una sentencia.

- Argumentos de función de tabla

  Como en los argumentos de funciones escalares, los argumentos de función de tabla pueden ser en general cualquier expresión SQL válida. Los siguientes ejemplos contienen sintaxis válida:

```
Ejemplo 1:  SELECT c1
            FROM TABLE( tf1('Zachary') ) AS z
            WHERE c2 = 'FLORIDA';

Ejemplo 2:  SELECT c1
            FROM TABLE( tf2 (:hostvar1, CURRENT DATE) ) AS z;

Ejemplo 3:  SELECT c1
              FROM t
              WHERE c2 IN
                       (SELECT c3 FROM
                        TABLE( tf5(t.c4) ) AS z  -- referencia correlacionada
                       )                         -- a cláusula FROM ant.
```

- Funciones de tabla que modifican datos de SQL

  Las funciones de tabla que se especifican con la opción MODIFIES SQL DATA sólo se pueden utilizar como la última referencia de tabla de una *sentencia-select*, *expresión-tabla-común* o sentencia RETURN que sea una subselección, una función SELECT INTO o una *selección completa de fila* de una sentencia SET. Sólo se permite una función de tabla en una cláusula FROM y los argumentos de la función de tabla deben estar correlacionados con las demás referencias de tabla

de la subselección (SQLSTATE 429BL). Los siguientes ejemplos contienen sintaxis válida para una función de tabla con la propiedad MODIFIES SQL DATA:

```
Ejemplo 1:  SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z

Ejemplo 2:  SELECT c1
            FROM t1, t2, TABLE( tfmod(t1.c1, t2.c1) ) AS z

Ejemplo 3:  SET var =
            (SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z

Ejemplo 4:  RETURN SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z

Ejemplo 5:  WITH v1(c1) AS
            (SELECT c1
            FROM TABLE( tfmod(:hostvar1) ) AS z)
            SELECT c1
            FROM v1, t1 WHERE v1.c1 = t1.c1
```

Referencias correlacionadas en referencias-tabla

Las referencias correlacionadas se pueden utilizar en expresiones de tabla anidadas o como argumentos para funciones de tabla. La regla básica que se aplica para ambos casos es que la referencia correlacionada debe ser de una *referencia-tabla* de un nivel superior en la jerarquía de subconsultas. Esta jerarquía incluye las referencias-tabla que ya se han resuelto en el proceso de izquierda a derecha de la cláusula FROM. En las expresiones de tabla anidadas, la palabra clave TABLE debe aparecer antes de la selección completa. Por lo tanto, los ejemplos siguientes son de sintaxis válida:

```
Ejemplo 1:  SELECT t.c1, z.c5
              FROM t, TABLE( tf3(t.c2) ) AS z      -- t precede tf3
              WHERE t.c3 = z.c4;                   -- en FROM, por lo que t.c2
                                                   -- se conoce

Ejemplo 2:  SELECT t.c1, z.c5
              FROM t, TABLE( tf4(2  * t.c2) ) AS z  -- t precede tf3
              WHERE t.c3 = z.c4;                   -- en FROM, por lo que t.c2
                                                   -- se conoce

Ejemplo 3:  SELECT d.deptno, d.deptname,
                    empinfo.avgsal, empinfo.empcount
              FROM department d,
                   TABLE (SELECT AVG(e.salary) AS avgsal,
                                 COUNT(*) AS empcount
                           FROM employee e           -- departamento precede
                           WHERE e.workdept=d.deptno -- y TABLE se ha
                          ) AS empinfo;              -- especificado, por lo que
                                                     -- d.deptno se conoce
```

Pero los ejemplos siguientes no son válidos:

```
Ejemplo 4:  SELECT t.c1, z.c5
              FROM TABLE( tf6(t.c2) ) AS z, t  -- no se puede resolver
                                                          t en t.c2!
              WHERE t.c3 = z.c4;                 -- compárese con el Ejemplo 1.

Ejemplo 5:  SELECT a.c1, b.c5
              FROM TABLE( tf7a(b.c2) ) AS a, TABLE( tf7b(a.c6) ) AS b
              WHERE a.c3 = b.c4;                  -- no se puede resolver b en
                                                  -- b.c2!
```

```
Ejemplo 6:  SELECT d.deptno, d.deptname,
                    empinfo.avgsal, empinfo.empcount
              FROM department d,
                   (SELECT AVG(e.salary) AS avgsal,
                               COUNT(*) AS empcount
                         FROM employee e           -- departamento precede
                         WHERE e.workdept=d.deptno -- pero TABLE no se ha
                        ) AS empinfo;              -- especificado, por lo que
                                                   -- d.deptno no se conoce
```

Referencias de tabla de cambio de datos

Una cláusula *referencia-tabla-cambio-datos* especifica una tabla de resultados intermedia. Esta tabla se basa en las filas que cambia directamente la sentencia UPDATE buscada, DELETE buscada o INSERT que se incluye en la cláusula. Una *referencia-tabla-cambio-datos* se puede especificar como la única *referencia-tabla* de la cláusula FROM de la selección completa externa que se utiliza en una *sentencia-select*, una sentencia SELECT INTO o una expresión de tabla común. También se puede especificar una *referencia-tabla-cambio-datos* como la única referencia de tabla en la única selección completa de una sentencia SET Variable (SQLSTATE 428FL). Se considera que la tabla o vista de destino de la sentencia de cambio de datos es una tabla o vista a la que se hace referencia en la consulta; por lo tanto, el ID de autorización de la consulta debe tener privilegio SELECT sobre la tabla o vista de destino.

El destino de la sentencia UPDATE, DELETE o INSERT no puede ser una vista temporal definida en una expresión de tabla común (SQLSTATE 42807).

FINAL TABLE
Especifica que las filas de la tabla de resultados intermedia representan el conjunto de filas que cambia la sentencia de cambio de datos de SQL tal como aparecen al final de la sentencia de cambio de datos. Si hay activadores AFTER o restricciones de referencia que dan lugar a más operaciones sobre la tabla que es el destino de la sentencia de cambio de datos de SQL, se devuelve un error (SQLSTATE 57058, SQLSTATE 560C6). Si el destino de la sentencia de cambio de datos de SQL es una vista que está definida con un activador INSTEAD OF para el tipo de cambio de datos, se devuelve un error (SQLSTATE 428G3).

NEW TABLE
Especifica que las filas de la tabla de resultados intermedia representan el conjunto de filas que cambia la sentencia de cambio de datos de SQL antes de la aplicación de restricciones de referencia y de activadores AFTER. Es posible que los datos de la tabla de destino al final de la sentencia no coincidan con los datos de la tabla de resultados intermedia debido al proceso adicional de restricciones de referencia y activadores AFTER.

OLD TABLE
Especifica que las filas de la tabla de resultados intermedia representan el conjunto de filas que cambia la sentencia de cambio de datos de SQL tal como aparecían antes de la aplicación de lasentencia de cambio de datos.

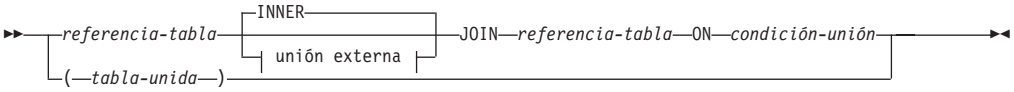
(sentencia-update-buscada)
Especifica una sentencia UPDATE buscada. Una cláusula WHERE o una cláusula SET en la sentencia UPDATE no puede contener referencias correlacionadas a columnas fuera de la sentencia UPDATE.

(sentencia-delete-buscada)
Especifica una sentencia DELETE buscada. Una cláusula WHERE en la sentencia DELETE no puede contener referencias correlacionadas a columnas fuera de la sentencia DELETE.

(sentencia-insert)
Especifica una sentencia INSERT. Una selección completa en la sentencia INSERT no puede contener referencias correlacionadas a columnas fuera de la selección completa de la sentencia INSERT.

El contenido de la tabla de resultados intermedia correspondiente a *referencia-tabla-cambio-datos* se determina cuando se abre el cursor. La tabla de resultados intermedia contiene todas las filas manipuladas, incluidas todas las columnas de la tabla o vista de destino especificada. Todas las columnas de la tabla o vista de destino correspondiente a una sentencia de cambio de datos SQL resultan accesibles utilizando los nombres de columnas de la tabla o vista de destino. Si se ha especificado una cláusula INCLUDE dentro de una sentencia de cambio de datos, la tabla de resultados intermedia contendrá estas columnas adicionales.

tabla-unida



unión externa:

Una *tabla unida* especifica una tabla resultante intermedia que es el resultado de una unión interna o una unión externa. La tabla se deriva de aplicar uno de los operadores de unión: INNER, LEFT OUTER, RIGHT OUTER o FULL OUTER a sus operandos.

Se puede decir que las uniones internas son el producto cruzado de las tablas (combinación de cada fila de la tabla izquierda con cada fila de la tabla derecha), conservando sólo las filas en que la condición de unión es verdadera. Es posible que a la tabla resultante le falten filas de una o ambas tablas unidas. Las uniones externas incluyen la unión interna y conservan las filas que faltan. Hay tres tipos de uniones externas:

- ***unión externa izquierda*** incluye las filas de la tabla de la izquierda que faltaban en la unión interna.
- ***unión externa derecha*** incluye las filas de la tabla de la derecha que faltaban en la unión interna.
- ***unión externa completa*** incluye las filas las tabla de la izquierda y de la derecha que faltaban en la unión interna.

Si no se especifica ningún operador-unión, el implícito es INNER. El orden en que se realizan múltiples uniones puede afectar al resultado. Las uniones pueden estar anidadas en otras uniones. El orden del proceso de uniones es generalmente de izquierda a derecha, pero se basa en la posición de la condición-unión necesaria. Es aconsejable utilizar paréntesis para que se pueda leer mejor el orden de las uniones anidadas. Por ejemplo:

```
TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1
  RIGHT JOIN TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1
    ON TB1.C1=TB3.C1
```

es igual a:

```
(TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1)
  RIGHT JOIN (TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1)
    ON TB1.C1=TB3.C1
```

Una tabla unida se puede utilizar en cualquier contexto en el que se utilice cualquier forma de la sentencia SELECT. Una vista o un cursor es de sólo lectura si su sentencia SELECT incluye una tabla unida.

Una *condición-unión* es una *condición-búsqueda* excepto en que:

- no puede contener ninguna subconsulta, escalar ni de cualquier otra clase
- no puede incluir ninguna operación de desreferencia ni una función DEREF en que el valor de referencia sea diferente del de la columna de identificador de objeto.
- no puede incluir una función SQL
- cualquier columna a la que una expresión haga referencia de la *condición-unión* debe ser una columna de una de las tablas de operandos de la unión asociada (en el ámbito de la misma cláusula tabla-unida)
- cualquier función a la que una expresión haga referencia de la *condición-unión* de una unión externa completa debe ser determinante y no tener ninguna acción externa.

Se produce un error si la condición de unión no cumple estas reglas (SQLSTATE 42972).

Las referencias a columnas se resuelven utilizando las reglas para la resolución de calificadores de nombres de columna. Las mismas reglas que se aplican a los predicados se aplican a las condiciones de unión.

Operaciones de unión

Una *condición-unión* especifica emparejamientos de T1 y T2, donde T1 y T2 son tablas de los operandos izquierdo y derecho del operador JOIN de la *condición-unión*. En todas las combinaciones posibles de filas de T1 y T2, una fila de T1 se empareja con una fila de T2 si la *condición-unión* es verdadera. Cuando una fila de T1 se une con una fila de T2, una fila del resultado consta de los valores de dicha fila de T1 concatenada con los valores de dicha fila de T2. La ejecución puede implicar la generación de una fila nula. La fila nula de una tabla consta de un valor nulo para cada columna de la tabla, sin tener en cuenta si las columnas permiten valores nulos.

A continuación encontrará un resumen del resultado de las operaciones de unión:

- El resultado de T1 INNER JOIN T2 consta de sus filas emparejadas cuando la condición-unión es verdadera.
- El resultado de T1 LEFT OUTER JOIN T2 consta de sus filas emparejadas cuando la condición-unión es verdadera y, para cada fila no emparejada de T1, la concatenación de dicha fila con la fila nula de T2. Todas las columnas derivadas de T2 permiten valores nulos.
- El resultado de T1 RIGHT OUTER JOIN T2 consta de sus filas emparejadas cuando la condición-unión es verdadera y, para cada fila de T2 no emparejada, la concatenación de dicha fila con la fila nula de T1. Todas las columnas derivadas de T1 permiten valores nulos.

- El resultado de T1 FULL OUTER JOIN T2 consta de sus filas emparejadas y, para cada fila de T2 no emparejada, la concatenación de dicha fila con la fila nula de T1 y, para cada fila de T1 no emparejada, la concatenación de dicha fila con la fila nula de T2. Todas las columnas derivadas de T1 y T2 permiten valores nulos.

cláusula-where

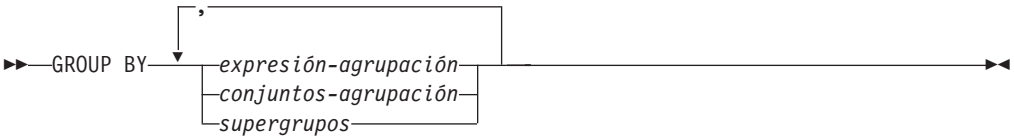
La cláusula WHERE especifica una tabla resultante intermedia que consta de aquellas filas de R para las que se cumple la *condición-búsqueda*. R es el resultado de la cláusula FROM de la subselección.

La *condición-búsqueda* debe ajustarse a las reglas siguientes:

- Cada *nombre-columna* debe identificar sin ambigüedades una columna de R o ser una referencia correlacionada. Un *nombre-columna* es una referencia correlacionada si identifica una columna de una *referencia-tabla* en una subselección externa.
- No debe especificarse una función de columna a menos que se especifique la cláusula WHERE en una subconsulta de una cláusula HAVING y el argumento de la función es una referencia correlacionada para un grupo.

Cualquier subconsulta de *condición-búsqueda* se ejecuta de forma efectiva para cada fila de R y los resultados se utilizan en la aplicación de la *condición-búsqueda* en la fila dada de R. Una subconsulta sólo se ejecuta en realidad para cada fila de R si incluye una referencia correlacionada. De hecho, una subconsulta sin referencias correlacionadas sólo se puede ejecutar una vez, mientras que una subconsulta con una referencia correlacionada puede tener que ejecutarse una vez para cada fila.

Cláusula group-by



La cláusula GROUP BY especifica una tabla resultante intermedia formada por una agrupación de las filas de R. R es el resultado de la cláusula anterior de la subselección.

En su forma más simple, una cláusula GROUP BY contiene una *expresión de agrupación*. Una expresión de agrupación es una *expresión* que se utiliza para definir la agrupación de R. Cada *nombre de columna* incluido en la expresión-agrupación debe identificar de forma inequívoca una columna de R (SQLSTATE 42702 o 42703). Una expresión de agrupación no puede incluir una selección completa escalar (SQLSTATE 42822) ni ninguna función que sea una variante o que tenga una acción externa (SQLSTATE 42845).

Las formas más complejas de la cláusula GROUP BY son los *conjuntos-agrupación* y los *supergrupos*. Para ver una descripción de estas formas, consulte los apartados "conjuntos-agrupación" y "supergrupos" en la página 486, respectivamente.

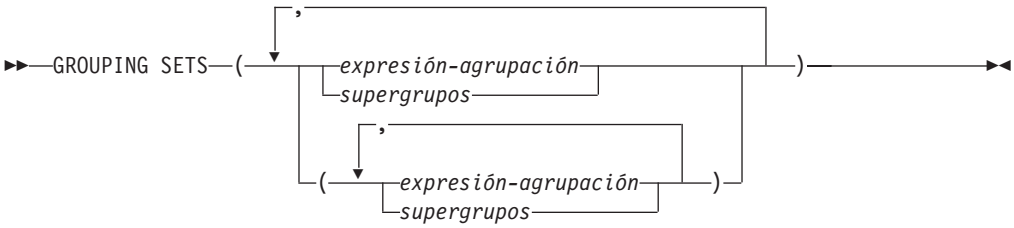
El resultado de GROUP BY es un conjunto de grupos de filas. Cada fila del resultado representa el conjunto de filas para el que la *expresión-agrupación* es igual. En la agrupación, todos los valores nulos de una *expresión-agrupación* se consideran iguales.

Una *expresión-agrupación* se puede utilizar en una condición de búsqueda de una cláusula HAVING, en una expresión de una cláusula SELECT o en una *expresión-clave-clasificación* de una cláusula ORDER BY (consulte el apartado "cláusula-order-by" en la página 491 para ver los detalles). En cada caso, la referencia sólo especifica un valor para cada grupo. Por ejemplo, si la *expresión-agrupación* es *col1+col2*, una expresión permitida en la lista de selección sería *col1+col2+3*. Las reglas de asociación para expresiones rechazarían la expresión parecida *3+col1+col2*, a menos que se utilicen paréntesis para asegurar que la expresión correspondiente se evalúe en el mismo orden. Por lo tanto, *3+(col1+col2)* también se permitiría en la lista de selección. Si se utiliza el operador de concatenación, la *expresión-agrupación* debe utilizarse exactamente como se ha especificado la expresión en la lista de selección.

Si la *expresión-agrupación* contiene series de longitud variable con blancos de cola, los valores del grupo pueden diferir en el número de blancos de cola y puede que no todos tengan la misma longitud. En dicho caso, la referencia a la *expresión-agrupación* continúa especificando sólo un valor para cada grupo, pero el valor para un grupo se elige arbitrariamente entre el conjunto de valores disponibles. Por lo tanto, la longitud real del valor del resultado es imprevisible.

Tal como se ha apuntado, existen casos en los que la cláusula GROUP BY no puede hacer referencia directamente a una columna que esté especificada en la cláusula SELECT como una expresión (selección completa-escalar, variante o funciones de acción externa). Para agrupar utilizando una expresión como ésta, utilice una expresión de tabla anidada o una expresión de tabla común para proporcionar primero una tabla resultante con la expresión como una columna del resultado. Para ver un ejemplo de la utilización de expresiones de tabla anidadas, consulte el "Ejemplo A9" en la página 496.

conjuntos-agrupación



Una especificación de *conjuntos-agrupación* permite especificar múltiples cláusulas de agrupación en una sola sentencia. Se puede decir que es la unión de dos o más grupos de filas en un solo conjunto resultante. Es lógicamente equivalente a la unión de múltiples subselecciones con la cláusula group by en cada subselección correspondiente a un conjunto de agrupación. Un conjunto de agrupación puede ser un solo elemento o puede ser una lista de elementos delimitados por

paréntesis, donde un elemento es una expresión-agrupación o un supergrupo. La utilización de *conjuntos-agrupación* permite calcular los grupos con una sola pasada por la tabla base.

La especificación de *conjuntos-agrupación* permite utilizar una *expresión-agrupación* simple o las formas más complejas de *supergrupos*. Para ver una descripción de *supergrupos*, consulte el apartado "supergrupos".

Tenga en cuenta que los conjuntos de agrupación son los bloques fundamentales para la creación de operaciones GROUP BY. Una operación GROUP BY simple con una sola columna puede considerarse un conjunto de agrupación con un elemento. Por ejemplo:

```
GROUP BY a
```

es igual a

```
GROUP BY GROUPING SETS((a))
```

y

```
GROUP BY a,b,c
```

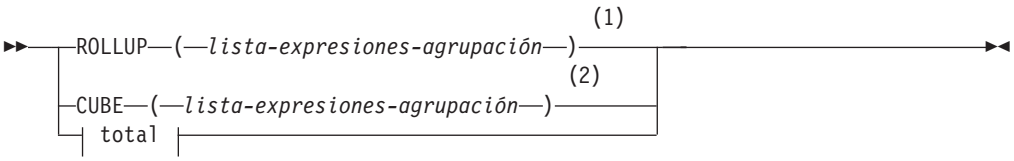
es igual a

```
GROUP BY GROUPING SETS((a,b,c))
```

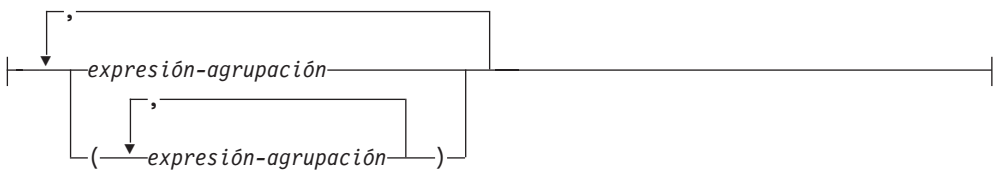
Las columnas de no agregación de la lista de selección de la subselección que se excluyen de un conjunto de agrupación devolverán un nulo para dichas columnas para cada fila generada para dicho conjunto de agrupación. Esto refleja el hecho que la agregación se ha realizado sin tener en cuenta los valores para dichas columnas.

Desde el "Ejemplo C2" en la página 500 al "Ejemplo C7" en la página 504 se ilustra la utilización de los conjuntos de agrupación.

supergrupos



lista-expresiones-agrupación:



total:

Notas:

1 Una especificación alternativa cuando se utiliza sola en la cláusula Group By es: lista-expresiones-agrupación WITH ROLLUP.

2 Una especificación alternativa cuando se utiliza sola en la cláusula Group By es: lista-expresiones-agrupación WITH CUBE.

ROLLUP (*lista-expresiones-agrupación* **)**

Una *agrupación ROLLUP* es una extensión de la cláusula GROUP BY que produce un conjunto resultante que contiene filas de *subtotales* además de las filas agrupadas "normales". Las filas de *subtotales* son filas "superagregadas" que contienen más agregrados cuyos valores se obtienen al aplicar las mismas funciones de columna que se han utilizado para obtener las filas agrupadas. Esta filas se llaman filas de subtotales porque ésta es su utilización más frecuente; sin embargo, puede utilizarse cualquier función de columna para la agregación. Por ejemplo, MAX y AVG se utilizan en "Ejemplo C8" en la página 506.

Una agrupación ROLLUP es una serie de *conjuntos-agrupación*. La especificación general de ROLLUP con *n* elementos

```
GROUP BY ROLLUP(C1,C2,...,Cn-1,Cn)
```

es equivalente a

```
GROUP BY GROUPING SETS((C1,C2,...,Cn-1,Cn)
                       (C1,C2,...,Cn-1)
                       ...
                       (C1,C2)
                       (C1)
                       () )
```

Observe que los *n* elementos de ROLLUP se convierten en *n*+1 conjuntos de agrupación. Tenga en cuenta también que el orden en el que se especifican las *expresiones-agrupación* es importante para ROLLUP. Por ejemplo:

```
GROUP BY ROLLUP(a,b)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b)
                       (a)
                       () )
```

mientras que

```
GROUP BY ROLLUP(b,a)
```

es igual a

```
GROUP BY GROUPING SETS((b,a)
                       (b)
                       () )
```

La cláusula ORDER BY es la única manera de garantizar el orden de las filas en el conjunto resultante. El "Ejemplo C3" en la página 501 ilustra la utilización de ROLLUP.

CUBE (*lista-expresiones-agrupación*)

Una *agrupación CUBE* es una extensión a la cláusula GROUP BY que produce un conjunto resultante que contiene todas las filas de la agregación ROLLUP y, además, contiene filas de "tabulación cruzada". Las filas de *tabulación cruzada* son filas "superagregadas" adicionales que no forman parte de una agregación con subtotales.

Igual que ROLLUP, una agrupación CUBE también puede decirse que es una serie de *conjuntos-agrupación*. En el caso de CUBE, todas las permutaciones de la *lista-expresiones-agregación* al cubo se calcula junto con el total. Por lo tanto, los *n* elementos de CUBE se convierten en 2**n* (2 elevado a la potencia *n*) *conjuntos-agrupación*. Por ejemplo, una especificación de

```
GROUP BY CUBE(a,b,c)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a,c)
                       (b,c)
                       (a)
                       (b)
                       (c)
                       () )
```

Observe que los 3 elementos de CUBE se convierten en 8 conjuntos de agrupaciones.

El orden de especificación de los elementos no importa para CUBE. 'CUBE (DayOfYear, Sales_Person)' y 'CUBE (Sales_Person, DayOfYear)' dan los mismos conjuntos del resultado. La utilización de la palabra 'mismos' se aplica al contenido del conjunto resultante, no a su orden. La cláusula ORDER BY es la única manera de garantizar el orden de las filas en el conjunto resultante. El "Ejemplo C4" en la página 501 ilustra la utilización de CUBE.

lista-expresiones-agrupación

Una *lista-expresiones-agrupación* se utiliza en la cláusula CUBE o ROLLUP para definir el número de elementos de la operación CUBE o ROLLUP. Se controla utilizando los paréntesis para delimitar los elementos con múltiples *expresiones-agrupación*.

Las reglas para la *expresión-agrupación* se describen en el apartado "Cláusula group-by" en la página 484. Por ejemplo, supongamos que una consulta tiene que devolver los gastos totales para ROLLUP de City dentro de una Province pero no de un County. Sin embargo la cláusula:

```
GROUP BY ROLLUP(Province, County, City)
```

da como resultado filas de subtotales que no se desean para County. En la cláusula

```
GROUP BY ROLLUP(Province, (County, City))
```

el compuesto (County, City) forma un elemento de ROLLUP y, por lo tanto, una consulta que utiliza esta cláusula dará el resultado deseado. En otras palabras, ROLLUP de dos elementos

```
GROUP BY ROLLUP(Province, (County, City))
```

genera

```
GROUP BY GROUPING SETS((Province, County, City)
                       (Province)
                       () )
```

mientras que ROLLUP de 3 elementos generaría

```
GROUP BY GROUPING SETS((Province, County, City)
                       (Province, County)
                       (Province)
                       () )
```

El “Ejemplo C2” en la página 500 también utiliza valores de columna compuestos.

total
Tanto CUBE como ROLLUP devuelven una fila que es la agregación global (total). Esto se puede especificar por separado mediante paréntesis vacíos dentro de la cláusula GROUPING SET. También puede especificarse directamente en la cláusula GROUP BY, aunque no surte ningún efecto en el resultado de la consulta. El “Ejemplo C4” en la página 501 utiliza la sintaxis del total.

Combinación de conjuntos de agrupaciones

Se puede utilizar para combinar cualquier tipo de cláusula GROUP BY. Cuando se combinan los campos de una *expresión-agrupación* simple con otros grupos, se ″añaden″ al principio de los *conjuntos de agrupación* resultantes. Cuando se combinan las expresiones ROLLUP o CUBE, funcionan como ″multiplicadores″ en el resto de la expresión, formando entradas de un conjunto de agrupación adicional según la definición de ROLLUP o CUBE.

Por ejemplo, la combinación de elementos de *expresión-agrupación* actúa de la siguiente manera:

```
GROUP BY a, ROLLUP(b,c)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a) )
```

O de manera parecida,

```
GROUP BY a, b, ROLLUP(c,d)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c,d)
                       (a,b,c)
                       (a,b) )
```

La combinación de elementos de *ROLLUP* actúa de la siguiente manera:

```
GROUP BY ROLLUP(a), ROLLUP(b,c)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a)
                       (b,c)
                       (b)
                       () )
```

De manera similar,

```
GROUP BY ROLLUP(a), CUBE(b,c)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a,c)
                       (a)
```

```
                        (b,c)
                        (b)
                        (c)
                        () )
```

La combinación de elementos de *CUBE* y de *ROLLUP* actúa de la siguiente manera:

```
GROUP BY CUBE(a,b), ROLLUP(c,d)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c,d)
                       (a,b,c)
                       (a,b)
                       (a,c,d)
                       (a,c)
                       (a)
                       (b,c,d)
                       (b,c)
                       (b)
                       (c,d)
                       (c)
                       () )
```

Igual que una *expresión-agrupación* simple, la combinación de conjuntos de agrupación elimina también los duplicados dentro de cada conjunto de agrupación. Por ejemplo,

```
GROUP BY a, ROLLUP(a,b)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b)
                       (a) )
```

Un ejemplo más completo de la combinación de conjuntos de agrupación es crear un conjunto resultante que elimine ciertas filas que se devolverían para una agregación CUBE completa.

Por ejemplo, considere la siguiente cláusula GROUP BY:

```
GROUP BY Region,
         ROLLUP(Sales_Person, WEEK(Sales_Date)),
         CUBE(YEAR(Sales_Date), MONTH (Sales_Date))
```

La columna listada inmediatamente a la derecha de GROUP BY está agrupada simplemente, las que están entre paréntesis a continuación de ROLLUP se han avanzado y las que están entre paréntesis a continuación de CUBE se han elevado al cubo. Por lo tanto, la cláusula anterior da como resultado el cubo de MONTH en YEAR que después avanza en WEEK en Sales_Person en la agregación Region. No da como resultado una fila del total ni ninguna fila de tabulación cruzada en Region, Sales_Person o WEEK(Sales_Date), por lo que produce menos filas que la cláusula siguiente:

```
GROUP BY ROLLUP (Region, Sales_Person, WEEK(Sales_Date),
                 YEAR(Sales_Date), MONTH(Sales_Date) )
```

cláusula-having

La cláusula HAVING especifica una tabla resultante intermedia que consta de aquellos grupos de R para los que se cumple la *condición-búsqueda*. R es el resultado de la cláusula anterior de la subselección. Si esta cláusula no es GROUP BY, R se considera un solo grupo sin columnas de agrupación.

Cada *nombre-columna* de la condición de búsqueda debe realizar una de las acciones siguientes:

- Identificar sin ambigüedades una columna de agrupación de R.
- Estar especificado dentro de la función de columna.
- Ser una referencia correlacionada. Un *nombre-columna* es una referencia correlacionada si identifica una columna de una *referencia-tabla* en una subselección externa.

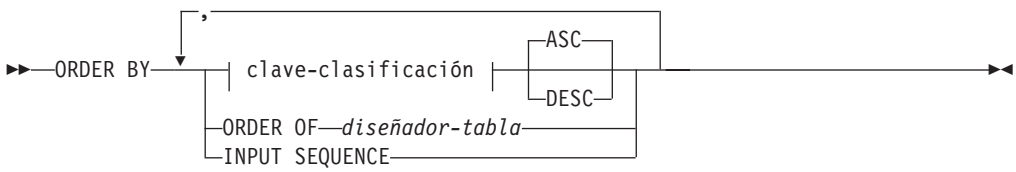
Un grupo de R al que se aplica la condición de búsqueda suministra el argumento para cada función de columna en la condición de búsqueda, excepto para cualquier función cuyo argumento sea una referencia correlacionada.

Si la condición de búsqueda contiene una subconsulta, puede considerarse que la subconsulta se ejecuta cada vez que se aplica la condición de búsqueda a un grupo de R y los resultados se utilizan en la aplicación de la condición de búsqueda. En realidad, la subconsulta sólo se ejecuta para cada grupo si contiene una referencia correlacionada. Para ver una ilustración de la diferencia, consulte el “Ejemplo A6” en la página 495 y el “Ejemplo A7” en la página 496.

Una referencia correlacionada a un grupo de R debe identificar una columna de agrupación o estar contenida en una función de columna.

Cuando se utiliza HAVING sin GROUP BY, la lista de selección sólo puede ser un nombre de columna dentro de una función de columna, una referencia correlacionada a columna, un literal o un registro especial.

cláusula-order-by



clave-clasificación:

La cláusula ORDER BY especifica una ordenación de las filas de la tabla resultante. Si se identifica una especificación de clasificación individual (una *clave-clasificación*

con una dirección asociada), las filas se ordenan por los valores de dicha especificación de clasificación. Si se indica más de una especificación de clasificación, las filas se ordenan por los valores de la primera especificación de clasificación identificada, después por los valores de la segunda especificación de clasificación identificada, y así sucesivamente. Ninguna *clave-clasificación* puede tener el tipo de LONG VARCHAR, CLOB, LONG VARGRAPHIC, DBCLOB, BLOB o DATALINK, un tipo diferenciado de cualquiera de estos tipos o un tipo estructurado (SQLSTATE 42907).

Una columna con nombre de la lista de selección se puede identificar mediante una *clave-clasificación* que sea un *entero-simple* o un *nombre-columna-simple*. Una columna sin nombre de la lista de selección debe identificarse por un *entero-simple* o, en algunos casos, por una *expresión-clave-clasificación* que coincida con la expresión de la lista de selección (consulte los detalles de *expresión-clave-clasificación*). Una columna no tiene nombre si no se especifica la cláusula AS y se obtiene a partir de una constante, una expresión con operadores o una función.

La ordenación se realiza de acuerdo con las reglas de comparación. El valor nulo es superior a cualquier otro valor. Si la cláusula ORDER BY no ordena por completo las filas, se visualizan las filas con valores duplicados de todas las columnas identificadas en un orden arbitrario.

nombre-columna-simple
Normalmente identifica una columna de la tabla resultante. En este caso, *nombre-columna-simple* debe ser el nombre de columna de una columna con nombre de la lista de selección.

El *nombre-columna-simple* también puede identificar un nombre de columna de una tabla, vista o tabla anidada identificada en la cláusula FROM si la consulta es una subselección. Se produce un error si la subselección:

- Especifica DISTINCT en la cláusula de selección (SQLSTATE 42822)
- Genera un resultado agrupado y el *nombre-columna-simple* no es una *expresión-agrupación* (SQLSTATE 42803).

La determinación de qué columna se utiliza para ordenar el resultado se describe más abajo en el apartado “Nombre de las columnas en claves de clasificación”.

entero-simple
Debe ser mayor que 0 y no ser superior al número de columnas de la tabla resultante (SQLSTATE 42805). El entero *n* identifica la columna *n* de la tabla resultante.

expresión-clave-clasificación
Una expresión que no es simplemente un nombre de columna ni una constante de enteros sin signo. La consulta a la que se aplica la ordenación debe ser una *subselección* para utilizar esta forma de clave-clasificación. La *expresión-clave-clasificación* no puede incluir una selección completa-escalar correlacionada (SQLSTATE 42703) ni una función con una acción externa (SQLSTATE 42845).

Cualquier nombre-columna de una *expresión-clave-clasificación* debe ajustarse a las reglas descritas bajo “Nombre de las columnas en claves de clasificación”.

Existen unos cuantos casos especiales que restringen más las expresiones que se pueden especificar.

- DISTINCT se especifica en la cláusula SELECT de la subselección (SQLSTATE 42822).

La expresión-clave-clasificación debe coincidir exactamente con una expresión de la lista de selección de la subselección (las selecciones completas-escalares nunca se emparejan).

- La subselección está agrupada (SQLSTATE 42803).

  La expresión-clave-clasificación puede:

  – ser una expresión en la lista de selección de la subselección,
  – incluir una *expresión-agrupación* de la cláusula GROUP BY de la subselección
  – incluir una función de columna, constante o variable del lenguaje principal.

ASC
Utiliza los valores de la columna en orden ascendente. Éste es el valor por omisión.

DESC
Utiliza los valores de la columna en orden descendente.

ORDER OF *diseñador-tabla*
Especifica que debe aplicarse el mismo orden utilizado en *diseñador-tabla* a la tabla resultante de la subselección. Debe haber una referencia de tabla que se corresponda con *diseñador-tabla* en la cláusula FROM de la subselección que especifica esta cláusula (SQLSTATE 42703). La subselección (o selección completa) correspondiente al *diseñador-tabla* especificado debe incluir una cláusula ORDER BY que dependa de los datos (SQLSTATE 428FI). El orden que se aplica es el mismo que si las columnas de la cláusula ORDER BY de la subselección anidada (o selección completa) se incluyeran en la subselección exterior (o selección completa) y estas columnas se especificaran en lugar de la cláusula ORDER OF.

Observe que este formato no se permite en una selección completa(excepto para el formato degenerativo de una selección completa). Por ejemplo, el ejemplo siguiente no es válido:

```
   (SELECT C1 FROM T1
   ORDER BY C1)
UNION
    SELECT C1 FROM T2
   ORDER BY ORDER OF T1
```

El ejemplo siguiente *sí* es válido:

```
SELECT C1 FROM
   (SELECT C1 FROM T1
UNION
    SELECT C1 FROM T2
    ORDER BY C1 ) AS UTABLE
ORDER BY ORDER OF UTABLE
```

INPUT SEQUENCE
Especifica que, para una sentencia INSERT, la tabla de resultados reflejará el orden de entrada de filas de datos ordenadas. El orden INPUT SEQUENCE sólo se puede especificar si se utiliza una sentencia INSERT en una cláusula FROM (SQLSTATE 428G4). Véase "referencia-tabla" en la página 475. Si se especifica INPUT SEQUENCE y los datos de entrada no están ordenados, la cláusula INPUT SEQUENCE se pasa por alto.

Notas:

- **Nombres de columna en claves de clasificación**:
  – El nombre de columna está calificado.

La consulta debe ser una *subselección* (SQLSTATE 42877). El nombre de columna debe identificar sin ambigüedades una columna de alguna tabla, vista o tabla anidada en la cláusula FROM de la subselección(SQLSTATE 42702). El valor de la columna se utiliza para calcular el valor de la especificación de clasificación.

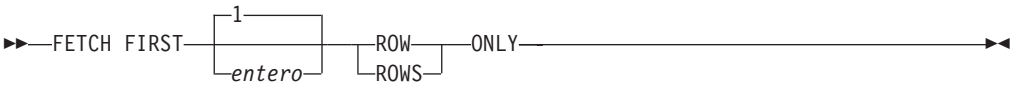
– El nombre de columna no está calificado.
  - La consulta es una subselección.

    Si el nombre de columna es idéntico al nombre de más de una columna de la tabla resultante, el nombre de columna debe identificar sin ambigüedades una columna de alguna tabla, vista o tabla anidada en la cláusula FROM de la subselección de orden (SQLSTATE 42702). Si el nombre de columna es idéntico a una columna, dicha columna se utiliza para calcular el valor de la especificación de clasificación. Si el nombre de columna no es idéntico a una columna de la tabla resultante, debe identificar sin ambigüedades una columna de alguna tabla, vista o tabla anidada en la cláusula FROM de la selección completa de la sentencia-select (SQLSTATE 42702).
  - La consulta no es una subselección (incluye operaciones de conjuntos como la unión, excepción o intersección).

    El nombre de columna no debe ser idéntico al nombre de más de una columna de la tabla resultante (SQLSTATE 42702). El nombre de columna debe ser idéntico a exactamente una columna de la tabla resultante (SQLSTATE 42707) y esta columna se utiliza para calcular el valor de la especificación de clasificación.

- **Límites**: La utilización de una *expresión-clave-clasificación* o un *nombre-columna-simple* donde la columna no está en la lista de selección puede dar como resultado la adición de la columna o expresión a la tabla temporal utilizada para clasificación. Esto puede dar como resultado que se alcance el límite del número de columnas de una tabla o el límite en el tamaño de una fila de una tabla. Si se exceden estos límites se producirá un error si es necesaria una tabla temporal para realizar la operación de clasificación.

cláusula-fetch-first



La *cláusula-fetch-first* establece el número máximo de filas que pueden recuperarse. Indica al gestor de bases de datos que la aplicación no recuperará más de *entero* filas, cualquiera que sea el número de filas que pueda haber en la tabla resultante cuando no se especifica esta cláusula. Cualquier intento de recuperar más filas que el número indicado por *entero* se trata de la misma manera que un fin de datos normal (SQLSTATE 02000). El valor de *entero* debe ser un entero positivo, distinto de cero.

Limitar la tabla resultante a las primeras *entero* filas puede mejorar el rendimiento. El gestor de bases de datos detendrá el proceso de la consulta una vez que haya determinado las *entero* primeras filas. Si se especifican la *cláusula-fetch-first* y la *cláusula-optimize-for*, se utiliza el valor *entero* más bajo de estas cláusulas para determinar el tamaño del almacenamiento intermedio de comunicaciones. Los valores se tienen en cuenta de forma independiente por motivos de optimización.

Si la selección completa contiene una sentencia de cambio de datos de SQL en la cláusula FROM, todas las filas se modifican, independientemente del límite del número de filas que hay que recuperar.

Ejemplos de subselecciones

Ejemplo A1: Seleccione todas las columnas y filas de la tabla EMPLOYEE.

```
SELECT * FROM EMPLOYEE
```

Ejemplo A2: Una las tablas EMP_ACT y EMPLOYEE, seleccione todas las columnas de la tabla EMP_ACT y añada el apellido del empleado (LASTNAME) de la tabla EMPLOYEE a cada fila del resultado.

```
SELECT EMP_ACT.*, LASTNAME
  FROM EMP_ACT, EMPLOYEE
  WHERE EMP_ACT.EMPNO = EMPLOYEE.EMPNO
```

Ejemplo A3: Una las tablas EMPLOYEE y DEPARTMENT, seleccione el número del empleado (EMPNO), el apellido del empleado (LASTNAME), el número del departamento (WORKDEPT en la tabla EMPLOYEE y DEPTNO en la tabla DEPARTMENT) y el nombre del departamento (DEPTNAME) de todos los empleados que han nacido (BIRTHDATE) con anterioridad a 1930.

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
   FROM EMPLOYEE, DEPARTMENT
   WHERE WORKDEPT = DEPTNO
   AND YEAR(BIRTHDATE) < 1930
```

Ejemplo A4: Seleccione el trabajo (JOB) y los salarios máximo y mínimo (SALARY) de cada grupo de filas con el mismo código de trabajo en la tabla EMPLOYEE, pero sólo para los grupos con más de una fila y con un salario máximo mayor o igual que 27000.

```
SELECT JOB, MIN(SALARY), MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY JOB
  HAVING COUNT(*) > 1
  AND MAX(SALARY) >= 27000
```

Ejemplo A5: Seleccione todas las filas de la tabla EMP_ACT para los empleados (EMPNO) del departamento (WORKDEPT) 'E11'. (Los números del departamento del empleado se muestran en la tabla EMPLOYEE.)

```
SELECT *
  FROM EMP_ACT
  WHERE EMPNO IN
           (SELECT EMPNO
               FROM EMPLOYEE
               WHERE WORKDEPT = 'E11')
```

Ejemplo A6: En la tabla EMPLOYEE, seleccione el número de departamento (WORKDEPT) y el salario (SALARY) máximo del departamento para todos los departamentos cuyo salario máximo sea menor que el salario medio de todos los empleados.

```
SELECT WORKDEPT, MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                           FROM EMPLOYEE)
```

La subconsulta de la cláusula HAVING sólo se ejecutará una vez en este ejemplo.

Ejemplo A7: Utilizando la tabla EMPLOYEE, seleccione el número de departamento (WORKDEPT) y el salario (SALARY) máximo del departamento para todos los departamentos cuyo salario máximo sea menor que el salario medio de los demás departamentos.

```
SELECT WORKDEPT, MAX(SALARY)
   FROM EMPLOYEE EMP_COR
    GROUP BY WORKDEPT
    HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                            FROM EMPLOYEE
                            WHERE NOT WORKDEPT = EMP_COR.WORKDEPT)
```

A diferencia del “Ejemplo A6” en la página 495, la subconsulta de la cláusula HAVING se habrá de ejecutar para cada grupo.

Ejemplo A8: Determine el número de empleado y el salario de los representantes de ventas junto con el salario medio y cuenta punta de sus departamentos.

Esta consulta primero debe crear una expresión de tabla anidada (DINFO) para obtener las columnas AVGSALARY y EMPCOUNT, así como la columna DEPTNO que se utiliza en la cláusula WHERE.

```
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY, DINFO.AVGSALARY, DINFO.EMPCOUNT
 FROM EMPLOYEE THIS_EMP,
       (SELECT OTHERS.WORKDEPT AS DEPTNO,
                        AVG(OTHERS.SALARY) AS AVGSALARY,
                        COUNT(*) AS EMPCOUNT
         FROM EMPLOYEE OTHERS
         GROUP BY OTHERS.WORKDEPT
        ) AS DINFO
 WHERE THIS_EMP.JOB = 'SALESREP'
 AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

Utilizar una expresión de tabla anidada para este caso ahorra la actividad general de crear la vista DIFO como una vista normal. Durante la preparación de la sentencia, se evita el acceso al catálogo para la vista y, debido al contexto del resto de la consulta, sólo se han de tener en cuenta las filas para el departamento de representantes de ventas para la vista.

Ejemplo A9: Visualice el nivel de formación medio y el salario de 5 grupos de empleados al azar.

Esta consulta necesita la utilización de una expresión de tabla anidada para establecer el valor aleatorio de cada empleado para que pueda utilizarse posteriormente en la cláusula GROUP BY.

```
SELECT RANDID , AVG(EDLEVEL), AVG(SALARY)
  FROM ( SELECT EDLEVEL, SALARY, INTEGER(RAND()*5) AS RANDID
           FROM EMPLOYEE
       ) AS EMPRAND
  GROUP BY RANDID
```

Ejemplo A10: Consultar la tabla EMP_ACT y devolver el número de los proyectos que tengan un empleado cuyo salario se encuentre entre los 10 más altos de todos los empleados.

```
SELECT EMP_ACT.EMPNO,PROJNO
   FROM EMP_ACT
   WHERE EMP_ACT.EMPNO IN
       (SELECT EMPLOYEE.EMPNO
        FROM EMPLOYEE
        ORDER BY SALARY DESC
        FETCH FIRST 10 ROWS ONLY)
```

Ejemplos de uniones

Ejemplo B1: Este ejemplo ilustra el resultado de varias uniones utilizando las tablas J1 y J2. Estas tablas contienen las filas que se muestran.

```
SELECT * FROM J1

W   X
--- ------
A       11
B       12
C       13

SELECT * FROM J2

Y   Z
--- ------
A       21
C       22
D       23
```

La siguiente consulta realiza una unión interna de J1 y J2, emparejando la primera columna de ambas tablas.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
```

En este ejemplo de unión interna, la fila con la columna W='C' de J1 y la fila con la columna Y='D' de J2 no se incluyen en el resultado porque no tienen una coincidencia en la otra tabla. Observe que la forma alternativa siguiente de una consulta de unión interna genera el mismo resultado.

```
SELECT * FROM J1, J2 WHERE W=Y
```

La unión externa izquierda siguiente recuperará la fila que falta de J1 con nulos para las columnas de J2. Se incluyen todas las filas de J1.

```
SELECT * FROM J1 LEFT OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
B       12 -        -
C       13 C       22
```

La siguiente unión externa derecha recuperará la fila que falta de J2 con nulos para las columnas de J1. Se incluyen todas la filas de J2.

```
SELECT * FROM J1 RIGHT OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
-        - D       23
```

La siguiente unión externa completa recuperará las filas que faltan de las dos tablas J1 y J2, con nulos cuando sea adecuado. Se incluyen todas las filas de las tablas J1 y J2.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
-        - D       23
B       12 -        -
```

Ejemplo B2: Utilizando las tablas J1 y J2 del ejemplo anterior, examine lo que pasa cuando se añade un predicado adicional a la condición de búsqueda.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=13

W   X      Y   Z
--- ------ --- ------
C       13 C       22
```

La condición adicional ha provocado que la unión interna sólo seleccionase 1 fila en comparación con la unión interna del “Ejemplo B1” en la página 497.

Observe el impacto que tiene en la unión externa completa.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=13

W   X      Y   Z
--- ------ --- ------
-        - A       21
C       13 C       22
-        - D       23
A       11 -        -
B       12 -        -
```

Ahora el resultado tiene 5 filas (a diferencia de 4 sin el predicado adicional) ya que sólo había 1 fila en la unión interna y tienen que devolverse todas las filas de ambas tablas.

La siguiente consulta ilustra que la colocación del mismo predicado adicional en la cláusula WHERE provoca resultados completamente diferentes.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
   WHERE X=13

W   X      Y   Z
--- ------ --- ------
C       13 C       22
```

La cláusula WHERE se aplica después del resultado intermedio de la unión externa completa. Este resultado intermedio sería el mismo que el resultado de la consulta de unión externa completa del “Ejemplo B1” en la página 497. La cláusula WHERE se aplica a este resultado intermedio y elimina todas las filas excepto la que contiene X=13. La elección de la ubicación de un predicado cuando se realizan uniones externas puede afectar significativamente a los resultados. Examine lo que pasa si el predicado es X=12 en lugar de X=13. La siguiente unión interna no devuelve ninguna fila.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=12
```

Por lo tanto, la unión externa completa devolvería 6 filas: 3 filas de J1 con nulos para las columnas de J2, y 3 filas de J2 con nulos para las columnas de J1.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=12

W   X      Y   Z
--- ------ --- ------
-         - A       21
-         - C       22
-         - D       23
A        11 -         -
B        12 -         -
C        13 -         -
```

En cambio, si el predicado adicional está en la cláusula WHERE, se devuelve la fila 1.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
   WHERE X=12

W   X      Y   Z
--- ------ --- ------
B        12 -         -
```

Ejemplo B3: Liste todos los departamentos con el número de empleado y el apellido del director, incluyendo los departamentos sin director.

```
SELECT DEPTNO, DEPTNAME, EMPNO, LASTNAME
  FROM DEPARTMENT LEFT OUTER JOIN EMPLOYEE
       ON MGRNO = EMPNO
```

Ejemplo B4: Liste todos los números de empleado y el apellido con el número de empleado y el apellido de su director, incluyendo los empleados sin director.

```
SELECT E.EMPNO, E.LASTNAME, M.EMPNO, M.LASTNAME
  FROM EMPLOYEE E LEFT OUTER JOIN
                                        DEPARTMENT INNER JOIN EMPLOYEE M
       ON MGRNO = M.EMPNO
       ON E.WORKDEPT = DEPTNO
```

La unión interna determina el apellido de cualquier director identificado en la tabla DEPARTMENT y la unión externa izquierda garantiza que se listen todos los empleados incluso si no se encuentra un departamento correspondiente en DEPARTMENT.

Ejemplos de conjuntos de agrupaciones, cube y rollup

Las consultas del "Ejemplo C1" en la página 500 al "Ejemplo C4" en la página 501 utilizan un subconjunto de filas de las tablas SALES basadas en el predicado 'WEEK(SALES_DATE) = 13'.

```
SELECT WEEK(SALES_DATE) AS WEEK,
      DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SALES AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
```

lo que da como resultado:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 LUCCHESSI                 3
         13           6 LUCCHESSI                 1
         13           6 LEE                       2
         13           6 LEE                       2
         13           6 LEE                       3
         13           6 LEE                       5
         13           6 GOUNOT                    3
         13           6 GOUNOT                    1
```

```
         13           6 GOUNOT                    7
         13           7 LUCCHESSI                 1
         13           7 LUCCHESSI                 2
         13           7 LUCCHESSI                 1
         13           7 LEE                       7
         13           7 LEE                       3
         13           7 LEE                       7
         13           7 LEE                       4
         13           7 GOUNOT                    2
         13           7 GOUNOT                   18
         13           7 GOUNOT                    1
```

Ejemplo C1: Esta es una consulta con una cláusula GROUP BY básica en 3 columnas:

```
SELECT WEEK(SALES_DATE) AS WEEK,
      DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
      SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

Da como resultado:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
         13           7 GOUNOT                   21
         13           7 LEE                      21
         13           7 LUCCHESSI                 4
```

Ejemplo C2: Genere el resultado basándose en dos conjuntos de agrupación diferentes de las filas de la tabla SALES.

```
SELECT WEEK(SALES_DATE) AS WEEK,
      DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
      SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY GROUPING SETS ( (WEEK(SALES_DATE), SALES_PERSON),
                         (DAYOFWEEK(SALES_DATE), SALES_PERSON))
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

Esto da como resultado:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           - GOUNOT                   32
         13           - LEE                      33
         13           - LUCCHESSI                 8
          -           6 GOUNOT                   11
          -           6 LEE                      12
          -           6 LUCCHESSI                 4
          -           7 GOUNOT                   21
          -           7 LEE                      21
          -           7 LUCCHESSI                 4
```

Las filas con WEEK 13 son del primer conjunto de agrupación y las demás filas son del segundo conjunto de agrupación.

Ejemplo C3: Si utiliza las 3 columnas diferenciadas implicadas en los conjuntos de agrupación del "Ejemplo C2" en la página 500 y lleva a cabo ROLLUP, puede ver los conjuntos de agrupación para (WEEK,DAY_WEEK,SALES_PERSON), (WEEK, DAY_WEEK), (WEEK) y el total.

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SUM(SALES) AS UNITS_SOLD
  FROM SALES
  WHERE WEEK(SALES_DATE) = 13
  GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
  ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

Esto da como resultado:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
         13           6 -                        27
         13           7 GOUNOT                   21
         13           7 LEE                      21
         13           7 LUCCHESSI                 4
         13           7 -                        46
         13           - -                        73
          -           - -                        73
```

Ejemplo C4: Si ejecuta la misma consulta que el "Ejemplo C3" sustituyendo sólo ROLLUP por CUBE, podrá ver conjuntos de agrupación adicionales para (WEEK,SALES_PERSON), (DAY_WEEK,SALES_PERSON), (DAY_WEEK), (SALES_PERSON) en el resultado.

```
SELECT WEEK(SALES_DATE) AS WEEK,
      DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
      SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY CUBE ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

Da como resultado:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
         13           6 -                        27
         13           7 GOUNOT                   21
         13           7 LEE                      21
         13           7 LUCCHESSI                 4
         13           7 -                        46
         13           - GOUNOT                   32
         13           - LEE                      33
         13           - LUCCHESSI                 8
         13           - -                        73
          -           6 GOUNOT                   11
          -           6 LEE                      12
          -           6 LUCCHESSI                 4
          -           6 -                        27
          -           7 GOUNOT                   21
          -           7 LEE                      21
          -           7 LUCCHESSI                 4
          -           7 -                        46
          -           - GOUNOT                   32
```

```
-          - LEE                    33
-          - LUCCHESSI               8
-          - -                      73
```

Ejemplo C5: Obtenga un conjunto resultante que incluya un total de filas seleccionadas de la tabla SALES junto con un grupo de filas agregadas por SALES_PERSON y MONTH.

```
SELECT SALES_PERSON,
       MONTH(SALES_DATE) AS MONTH,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY GROUPING SETS ( (SALES_PERSON, MONTH(SALES_DATE)),
                         ()
                       )
ORDER BY SALES_PERSON, MONTH
```

Esto da como resultado:

```
SALES_PERSON    MONTH       UNITS_SOLD
--------------- ----------- -----------
GOUNOT                    3          35
GOUNOT                    4          14
GOUNOT                   12           1
LEE                       3          60
LEE                       4          25
LEE                      12           6
LUCCHESSI                 3           9
LUCCHESSI                 4           4
LUCCHESSI                12           1
          - -                       155
```

Ejemplo C6: Este ejemplo muestra dos consultas ROLLUP simples seguidas de una consulta que trata los dos ROLLUP como conjuntos de agrupación en un sólo conjunto resultante y especifica el orden de filas para cada columna implicada en los conjuntos de agrupación.

Ejemplo C6-1:

```
SELECT WEEK(SALES_DATE) AS WEEK,
      DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) )
ORDER BY WEEK, DAY_WEEK
```

da como resultado:

```
WEEK        DAY_WEEK    UNITS_SOLD
----------- ----------- -----------
         13           6          27
         13           7          46
         13           -          73
         14           1          31
         14           2          43
         14           -          74
         53           1           8
         53           -           8
          -           -         155
```

Ejemplo C6-2:

```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP ( MONTH(SALES_DATE), REGION );
ORDER BY MONTH, REGION
```

da como resultado:

```
MONTH       REGION          UNITS_SOLD
----------- --------------- -----------
          3 Manitoba                 22
          3 Ontario-North             8
          3 Ontario-South            34
          3 Quebec                   40
          3 -                       104
          4 Manitoba                 17
          4 Ontario-North             1
          4 Ontario-South            14
          4 Quebec                   11
          4 -                        43
         12 Manitoba                  2
         12 Ontario-South             4
         12 Quebec                    2
         12 -                         8
          - -                       155
```

Ejemplo C6-3:

```
SELECT WEEK(SALES_DATE) AS WEEK,
      DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY GROUPING SETS ( ROLLUP( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) ),
                         ROLLUP( MONTH(SALES_DATE), REGION )  )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

da como resultado:

```
WEEK        DAY_WEEK    MONTH       REGION          UNITS_SOLD
----------- ----------- ----------- --------------- -----------
         13           6           - -                        27
         13           7           - -                        46
         13           -           - -                        73
         14           1           - -                        31
         14           2           - -                        43
         14           -           - -                        74
         53           1           - -                         8
         53           -           - -                         8
          -           -           3 Manitoba                 22
          -           -           3 Ontario-North             8
          -           -           3 Ontario-South            34
          -           -           3 Quebec                   40
          -           -           3 -                       104
          -           -           4 Manitoba                 17
          -           -           4 Ontario-North             1
          -           -           4 Ontario-South            14
          -           -           4 Quebec                   11
          -           -           4 -                        43
          -           -          12 Manitoba                  2
          -           -          12 Ontario-South             4
          -           -          12 Quebec                    2
```

```
-          -          12 -                      8
-          -           - -                    155
-          -           - -                    155
```

La utilización de los dos ROLLUP como conjuntos de agrupación hace que el resultado incluya filas duplicadas. Incluso hay dos filas del total.

Observe cómo la utilización de ORDER BY ha afectado al resultado:

- En el primer conjunto agrupado, la semana 53 se ha cambiado a la posición final.
- En el segundo conjunto agrupado, el mes 12 se ha puesto al final y las regiones aparecen por orden alfabético.
- Los valores nulos se clasifican arriba.

Ejemplo C7: En las consultas que realizan varios ROLLUP en una sola pasada (como por ejemplo, "Ejemplo C6-3" en la página 503) tiene la posibilidad de indicar, si lo desea, qué conjunto de agrupación ha producido cada fila. Los pasos siguientes demuestran cómo proporcionar una columna (denominada GROUP) que indica el origen de cada fila del conjunto resultante. Por origen se quiere decir cual de los dos conjuntos de agrupación ha producido la fila del conjunto resultante.

Paso 1: Introduzca una manera de ″generar″ los nuevos valores de datos, utilizando una consulta que los selecciona en la cláusula VALUES (que es una forma alternativa de una selección completa). Esta consulta muestra cómo se puede derivar una tabla llamada ″X″ que tienen 2 columnas ″R1″ y ″R2″ y 1 fila de datos.

```
SELECT R1,R2
FROM (VALUES('GROUP 1','GROUP 2')) AS X(R1,R2);
```

da como resultado:

```
  R1      R2
------- -------
GROUP 1 GROUP 2
```

Paso 2: Forme el producto cruzado de esta tabla ″X″ con la tabla SALES. Esto añade las columnas ″R1″ y ″R2″ a cada fila.

```
SELECT R1, R2, WEEK(SALES_DATE) AS WEEK,
      DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SALES AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
```

Esto añade las columnas ″R1″ y ″R2″ a cada fila.

Paso 3: Ahora se pueden combinar estas columnas con los conjuntos de agrupación para que incluyan estas columnas en el análisis de avance.

```
SELECT R1, R2,
       WEEK(SALES_DATE) AS WEEK,
      DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION, SUM(SALES) AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
GROUP BY GROUPING SETS ((R1, ROLLUP(WEEK(SALES_DATE),
                         DAYOFWEEK(SALES_DATE))),
                       (R2,ROLLUP( MONTH(SALES_DATE), REGION ) )  )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

da como resultado:

```
R1      R2      WEEK     DAY_WEEK  MONTH     REGION          UNITS_SOLD
------- ------- -------- --------- --------- --------------- -----------
GROUP 1 -                13           6           - -                27
GROUP 1 -                13           7           - -                46
GROUP 1 -                13           -           - -                73
GROUP 1 -                14           1           - -                31
GROUP 1 -                14           2           - -                43
GROUP 1 -                14           -           - -                74
GROUP 1 -                53           1           - -                 8
GROUP 1 -                53           -           - -                 8
-       GROUP 2           -           -           3 Manitoba         22
-       GROUP 2           -           -           3 Ontario-North     8
-       GROUP 2           -           -           3 Ontario-South    34
-       GROUP 2           -           -           3 Quebec           40
-       GROUP 2           -           -           3 -               104
-       GROUP 2           -           -           4 Manitoba         17
-       GROUP 2           -           -           4 Ontario-North     1
-       GROUP 2           -           -           4 Ontario-South    14
-       GROUP 2           -           -           4 Quebec           11
-       GROUP 2           -           -           4 -                43
-       GROUP 2           -           -          12 Manitoba          2
-       GROUP 2           -           -          12 Ontario-South     4
-       GROUP 2           -           -          12 Quebec            2
-       GROUP 2           -           -          12 -                 8
-       GROUP 2           -           -           - -               155
GROUP 1 -                 -           -           - -               155
```

Paso 4: Tenga en cuenta que puesto que R1 y R2 se utilizan en conjuntos de agrupación diferentes, siempre que R1 no sea nulo en el resultado, R2 es nulo y siempre que R2 sea no nulo en el resultado, R1 es nulo. Esto significa que puede consolidar estas columnas en una sola utilizando la función COALESCE. También puede utilizar esta columna en la cláusula ORDER BY para conservar el resultado de los dos conjuntos de agrupación juntos.

```
SELECT COALESCE(R1,R2) AS GROUP,
       WEEK(SALES_DATE) AS WEEK,
      DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION, SUM(SALES) AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
GROUP BY GROUPING SETS ((R1, ROLLUP(WEEK(SALES_DATE),
                         DAYOFWEEK(SALES_DATE))),
                       (R2,ROLLUP( MONTH(SALES_DATE), REGION ) )  )
ORDER BY GROUP, WEEK, DAY_WEEK, MONTH, REGION;
```

da como resultado:

```
GROUP   WEEK        DAY_WEEK    MONTH       REGION          UNITS_SOLD
------- ----------- ----------- ----------- --------------- -----------
GROUP 1          13           6           - -                        27
GROUP 1          13           7           - -                        46
GROUP 1          13           -           - -                        73
GROUP 1          14           1           - -                        31
GROUP 1          14           2           - -                        43
GROUP 1          14           -           - -                        74
GROUP 1          53           1           - -                         8
GROUP 1          53           -           - -                         8
GROUP 1           -           -           - -                       155
GROUP 2           -           -           3 Manitoba                 22
GROUP 2           -           -           3 Ontario-North             8
GROUP 2           -           -           3 Ontario-South            34
GROUP 2           -           -           3 Quebec                   40
GROUP 2           -           -           3 -                       104
GROUP 2           -           -           4 Manitoba                 17
GROUP 2           -           -           4 Ontario-North             1
GROUP 2           -           -           4 Ontario-South            14
GROUP 2           -           -           4 Quebec                   11
```

```
GROUP 2          -          -          4 -                       43
GROUP 2          -          -         12 Manitoba                 2
GROUP 2          -          -         12 Ontario-South            4
GROUP 2          -          -         12 Quebec                   2
GROUP 2          -          -         12 -                        8
GROUP 2          -          -          - -                      155
```

Ejemplo C8: El ejemplo siguiente ilustra la utilización de varias funciones de columna cuando se realiza un CUBE. El ejemplo también utiliza funciones de conversión del tipo de datos y el redondeo para producir resultados decimales con una precisión y escala razonables.

```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD,
       MAX(SALES) AS BEST_SALE,
       CAST(ROUND(AVG(DECIMAL(SALES)),2) AS DECIMAL(5,2)) AS AVG_UNITS_SOLD
FROM SALES
GROUP BY CUBE(MONTH(SALES_DATE),REGION)
ORDER BY MONTH, REGION
```

Da como resultado:

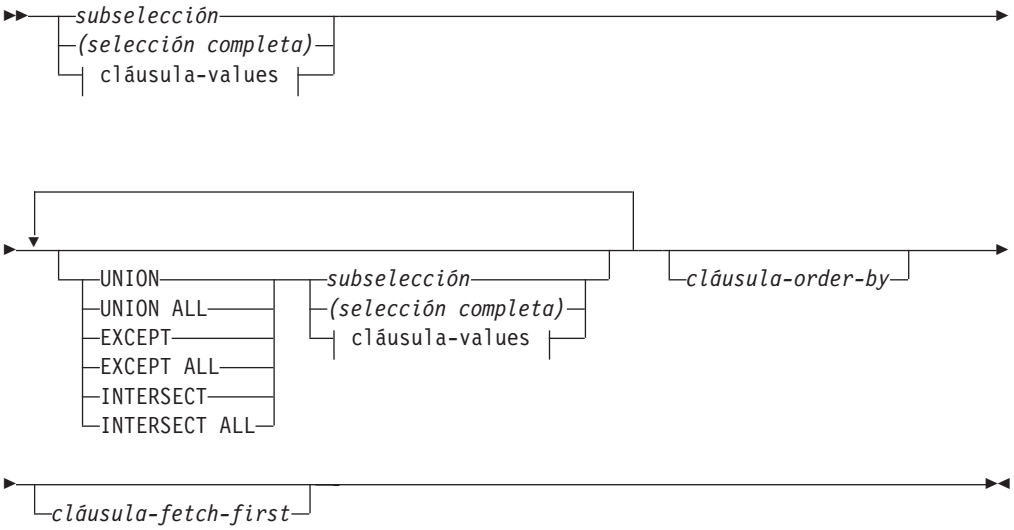
```
MONTH       REGION          UNITS_SOLD  BEST_SALE   AVG_UNITS_SOLD
----------- --------------- ----------- ----------- --------------
          3 Manitoba                 22           7           3.14
          3 Ontario-North             8           3           2.67
          3 Ontario-South            34          14           4.25
          3 Quebec                   40          18           5.00
          3 -                       104          18           4.00
          4 Manitoba                 17           9           5.67
          4 Ontario-North             1           1           1.00
          4 Ontario-South            14           8           4.67
          4 Quebec                   11           8           5.50
          4 -                        43           9           4.78
         12 Manitoba                  2           2           2.00
         12 Ontario-South             4           3           2.00
         12 Quebec                    2           1           1.00
         12 -                         8           3           1.60
          - Manitoba                 41           9           3.73
          - Ontario-North             9           3           2.25
          - Ontario-South            52          14           4.00
          - Quebec                   53          18           4.42
          - -                       155          18           3.87
```

Información relacionada:

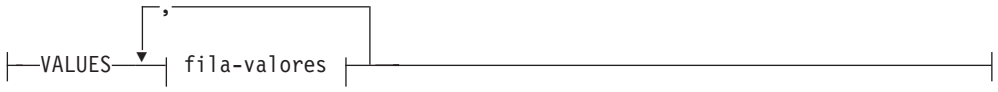
- "Identificadores" en la página 64
- "Funciones" en la página 164
- "GROUPING" en la página 281
- "Selección completa" en la página 508
- "Sentencia select" en la página 513
- "Sentencia DELETE" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia INSERT" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia UPDATE" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia CREATE FUNCTION (Escalar de SQL, tabla o fila)" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia CREATE FUNCTION (Tabla externa)" en la publicación *Consulta de SQL, Volumen 2*
- "Series de caracteres" en la página 90
- "Asignaciones y comparaciones" en la página 110

• “Predicados” en la página 219

Selección completa



cláusula-values:



fila-valores:

La *selección completa* es un componente de la sentencia select, la sentencia INSERT y la sentencia CREATE VIEW. También es un componente de ciertos predicados que, a su vez, son componentes de una sentencia. Una selección completa que es un componente de un predicado se denomina una *subconsulta* y una selección completa que vaya entre paréntesis a veces se denomina también una subconsulta.

Los operadores de conjuntos UNION, EXCEPT e INTERSECT se corresponden a los operadores relacionales de unión, diferencia e intersección.

Una selección completa especifica una tabla resultante. Si no se utiliza un operador de conjunto, el resultado de la selección completa es el resultado de la subselección especificada o la cláusula-values.

cláusula-values
Obtiene una tabla resultante especificando los valores reales, utilizando expresiones, para cada columna de una fila de la tabla resultante. Pueden especificarse múltiples filas.

NULL sólo se puede utilizar con múltiples especificaciones de *fila-valores* y como mínimo una fila de la misma columna no debe ser NULL (SQLSTATE 42826).

Una *fila-valores* se especifica por:

- Una sola expresión para una tabla resultante de una sola columna o
- *n* expresiones (o NULL) separadas por comas y encerradas entre paréntesis, donde *n* es el número de columnas de la tabla resultante.

La cláusula VALUES de múltiples filas deben tener el mismo número de expresiones en cada *fila-valores* (SQLSTATE 42826).

A continuación encontrará ejemplos de cláusulas-values y su significado.

```
VALUES (1),(2),(3)          - 3 filas de 1 columna
VALUES 1, 2, 3              - 3 filas de 1 columna
VALUES (1, 2, 3)            - 1 fila de 3 columnas
VALUES (1,21),(2,22),(3,23) - 3 filas de 2 columnas
```

Una cláusula-values que está compuesta de *n* especificaciones de *fila-valores*, RE_1 a RE_n, donde *n* es mayor que 1, es equivalente a:

RE_1 UNION ALL RE_2 ... UNION ALL RE_n

Esto significa que las expresiones correspondientes de cada *fila-valores* deben ser comparables (SQLSTATE 42825).

UNION o **UNION ALL**
Obtiene una tabla resultante combinando las otras dos tablas resultantes (R1 y R2). Si se ha especificado UNION ALL, el resultado consta de todas las filas de R1 y de R2. Si se especifica UNION sin la opción ALL, el resultado consta de todas las filas de R1 o R2, con las filas duplicadas eliminadas. Sin embargo, en cualquier caso, todas las filas de la tabla UNION es una fila de R1 o una fila de R2.

EXCEPT o **EXCEPT ALL**
Obtiene una tabla resultante combinando las otras dos tablas resultantes (R1 y R2). Si se especifica EXCEPT ALL, el resultado consta de todas las filas que no tienen una fila correspondiente en R2, donde las filas duplicadas son significativas. Si se especifica EXCEPT sin la opción ALL, el resultado consta de todas las filas que están sólo en R1, y las filas duplicadas se eliminan del resultado de esta operación.

INTERSECT o **INTERSECT ALL**
Obtiene una tabla resultante combinando las otras dos tablas resultantes (R1 y R2). Si se especifica INTERSECT ALL, el resultado consta de todas las filas que están en R1 y en R2. Si se especifica INTERSECT sin la opción ALL, el resultado consta de todas las filas que están en R1 y en R2, con las filas duplicadas eliminadas.

cláusula-order-by
Una selección completa que contenga una cláusula ORDER BY o FETCH FIRST no puede especificarse en:

- Una tabla de consulta materializada
- La selección completa más externa de una vista (SQLSTATE 428FJ).

Nota: Una cláusula ORDER BY en una selección completa no afecta el orden de las filas que una consulta devuelve. Una cláusula ORDER BY sólo afecta el orden de las filas devueltas si se especifica en la selección completa más externa.

El número de columnas de las tablas resultantes R1 y R2 han de ser iguales (SQLSTATE 42826). Si no se especifica la palabra clave ALL, R1 y R2 no deben contener ninguna columna que tenga el tipo de datos LONG VARCHAR, CLOB, LONG VARGRAPHIC, DBCLOB, BLOB, DATALINK, un tipo diferenciado de estos tipos o un tipo estructurado (SQLSTATE 42907).

Las columnas del resultado tienen estos nombres:

- Si las columnas *n* de R1 y *n* de R2 tienen el mismo nombre de columna del resultado, la columna *n* de R tiene el nombre de la columna del resultado.
- Si la columna *n* de R1 y la columna *n* de R2 tienen nombres de columna del resultado diferente, se genera un nombre. Este nombre no se puede utilizar como nombre de columna en una cláusula ORDER BY o UPDATE.

El nombre generado se puede determinar ejecutando DESCRIBE de la sentencia de SQL y consultando el campo SQLNAME.

Dos filas se consideran duplicadas si cada valor de la primera es igual al valor correspondiente de la segunda. (Para la determinación de duplicados, dos valores nulos se consideran iguales.)

Cuando se combinan múltiples operaciones en una expresión, las operaciones entre paréntesis se llevan a cabo primero. Si no hay paréntesis, las operaciones se llevan a cabo de izquierda a derecha a excepción de que todas las operaciones INTERSECT se efectúan antes que las operaciones UNION o EXCEPT.

En el ejemplo siguiente, los valores de las tablas R1 y R2 se muestran en la izquierda. Las otras cabeceras listadas muestran los valores como resultado de varias operaciones de conjunto en R1 y en R2.

R1	R2	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTER-SECT ALL	INTER-SECT
1	1	1	1	1	2	1	1
1	1	1	2	2	5	1	3
1	3	1	3	2		3	4
2	3	1	4	2		4	
2	3	1	5	4			
2	3	2		5			
3	4	2					
4		2					
4		3					
5		3					
		3					
		3					
		3					
		4					
		4					
		4					
		5					

Ejemplos de una selección completa

Ejemplo 1: Seleccione todas las columnas y filas de la tabla EMPLOYEE.

```
SELECT * FROM EMPLOYEE
```

Ejemplo 2: Liste los números de empleado (EMPNO) de todos los empleados de la tabla EMPLOYEE cuyo número de departamento (WORKDEPT) empiece por 'E' **o** que estén asignados a proyectos de la tabla EMP_ACT cuyo número de proyecto (PROJNO) sea igual a 'MA2100', 'MA2110' o 'MA2112'.

```
SELECT EMPNO
   FROM EMPLOYEE
   WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

Ejemplo 3: Haga la misma consulta que en el ejemplo 2 y, además, “identifique” las filas de la tabla EMPLOYEE con 'emp' y las filas de la tabla EMP_ACT con 'emp_act'. A diferencia del resultado del ejemplo 2, esta consulta puede devolver el mismo EMPNO más de una vez, identificando la tabla del que proviene mediante el “identificador” asociado.

```
SELECT EMPNO, 'emp'
   FROM EMPLOYEE
   WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO, 'emp_act' FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

Ejemplo 4: Realice la misma consulta que en el ejemplo 2, sólo que utilice UNION ALL para que no se elimine ninguna fila duplicada.

```
SELECT EMPNO
    FROM EMPLOYEE
    WHERE WORKDEPT LIKE 'E%'
UNION ALL
SELECT EMPNO
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

Ejemplo 5: Realice la misma consulta que en el ejemplo 3, sólo que esta vez incluya dos empleados adicionales que no están actualmente en ninguna tabla e identifíquelos como ″new″.

```
SELECT EMPNO, 'emp'
   FROM EMPLOYEE
   WHEREWORKDEPTLIKE 'E%'
UNION
SELECT EMPNO, 'emp_act'
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
UNION
   VALUES ('NEWAAA', 'new'), ('NEWBBB', 'new')
```

Ejemplo 6: Este ejemplo de EXCEPT genera todas las filas que están en T1 pero no en T2.

```
(SELECT * FROM T1)
EXCEPT ALL
(SELECT * FROM T2)
```

Si no hay ningún valor NULL implicado, este ejemplo devuelve los mismos resultados que

```
SELECT ALL *
   FROM T1
   WHERE NOT EXISTS (SELECT * FROM T2
                WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

Ejemplo 7: Este ejemplo de INTERSECT genera todas las filas que están en ambas tablas, T1 y T2, eliminando los duplicados.

```
(SELECT * FROM T1)
INTERSECT
(SELECT * FROM T2)
```

Si no hay valores NULL implicados, este ejemplo devuelve el mismo resultado que

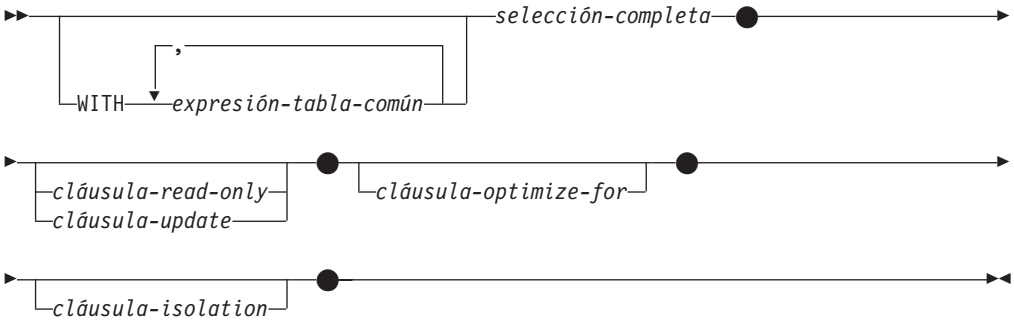
```
SELECT DISTINCT * FROM T1
   WHERE EXISTS (SELECT * FROM T2
                WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

donde C1, C2, etcétera representan las columnas de T1 y T2.

Información relacionada:

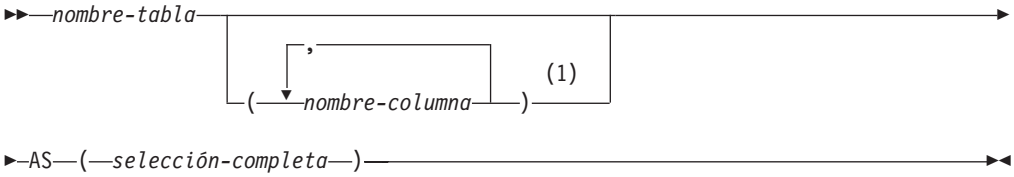
- "Reglas para los tipos de datos del resultado" en la página 125
- "Reglas para la conversión de series" en la página 129

Sentencia select



La *sentencia-select* es la forma de una consulta que puede especificarse directamente en una sentencia DECLARE CURSOR o prepararse y después hacerse referencia en una sentencia DECLARE CURSOR. También puede emitirse mediante sentencias de SQL dinámico utilizando el procesador de la línea de mandatos (o herramientas similares) haciendo que se visualice una tabla resultante en la pantalla del usuario. En cualquier caso, la tabla especificada por una *sentencia-select* es el resultado de la selección completa.

expresión-común-tabla



Notas:

1 Si una expresión de tabla común es recursiva o si la selección completa da como resultado nombres de columna duplicados, deben especificarse los nombres de columna.

Una *expresión de tabla común* permite la definición de una tabla resultante con un *nombre-tabla* que puede especificarse como un nombre de tabla en cualquier cláusula FROM de la selección completa que sigue. Se pueden especificar múltiples expresiones de tabla comunes a continuación de una sola palabra clave WITH. Cada expresión de tabla común especificada también puede referirse por nombre en la cláusula FROM de expresiones de tabla comunes subsiguientes.

Si se especifica una lista de columnas, debe constar de tantos nombres como el número de columnas de la tabla resultante de la selección completa. Cada *nombre-columna* debe ser exclusivo y no calificado. Si no se especifican estos nombres de columna, los nombres se obtienen de la lista de selección de la selección completa utilizada para definir la expresión de tabla común.

El *nombre-tabla* de una expresión de tabla común debe ser diferente de cualquier otro *nombre-tabla* de una expresión de tabla común de la misma sentencia (SQLSTATE 42726). Si se especifica la expresión de tabla común en una sentencia INSERT, el *nombre-tabla* no puede ser el mismo que el nombre de tabla o vista que es el objeto de la inserción (SQLSTATE 42726). Un *nombre-tabla* de una expresión de

tabla común puede especificarse como nombre de tabla en cualquier cláusula FROM de toda la selección completa. Un *nombre-tabla* de una expresión de tabla común prevalece sobre cualquier tabla, vista o seudónimo existentes (en el catálogo) que tenga el mismo nombre calificado.

Si se define más de una expresión de tabla común en la misma sentencia, no están permitidas las referencias cíclicas entre las expresiones de tabla comunes (SQLSTATE 42835). Una *referencia cíclica* se produce cuando dos expresiones de tabla comunes *dt1* y *dt2* están creadas de tal manera que *dt1* hace referencia a *dt2* y *dt2* hace referencia a *dt1*.

Si una selección completa de una expresión de tabla común contiene una *referencia a tabla de cambio de datos* en la cláusula FROM, se dice que la expresión de tabla común modifica los datos. Una expresión de tabla común que modifica datos siempre se evalúa cuando se procesa la sentencia, independientemente de si la expresión de tabla común se utiliza en cualquier otro lugar de la sentencia. Si hay al menos una expresión de tabla común que lee o modifica datos, todas las expresiones de tabla común se procesan en el orden en el que aparecen, y cada expresión de tabla común que lee o modifica datos se ejecuta por completo, incluidas todas las restricciones y activadores, antes de que se ejecuten las expresiones de tabla común siguientes.

La expresión de tabla común también es opcional antes de la selección completa en las sentencias CREATE VIEW e INSERT.

Una expresión de tabla común puede utilizarse:

- En lugar de una vista para evitar crear la vista (cuando no sea necesaria la utilización general de la vista y no se utilicen actualizaciones ni supresiones colocadas)
- Para permitir la agrupación por una columna que se obtiene de una subselección o función escalar que no es determinista o tiene una acción externa
- Cuando la tabla resultante deseada se basa en variables del lenguaje principal
- Cuando la misma tabla resultante necesite compartirse en una selección completa
- Cuando el resultado necesita obtenerse mediante recurrencia
- Cuando se tienen que procesar varias sentencias de cambio de datos SQL dentro de la consulta

Si la selección completa de una expresión de tabla común contiene una referencia a sí misma en una cláusula FROM, la expresión de tabla común es *recursiva*. Las consultas que utilizan la recurrencia son útiles en las aplicaciones que permiten su uso, tales como la lista de material (BOM), sistemas de reservas y planificación de la red.

Deben cumplirse las condiciones siguientes en una expresión de tabla común recursiva:

- Cada selección completa que forma parte del ciclo de repetición debe empezar por SELECT o SELECT ALL. La utilización de SELECT DISTINCT no está permitida (SQLSTATE 42925). Además, las uniones deben utilizar UNION ALL (SQLSTATE 42925).
- Los nombres de columna deben especificarse a continuación del *nombre-tabla* de la expresión de tabla común (SQLSTATE 42908).

- La primera selección completa de la primera unión (la selección completa de inicialización) no debe incluir ninguna referencia a ninguna columna de la expresión de tabla común de cualquier cláusula FROM (SQLSTATE 42836).
- Si se hace referencia a un nombre de columna de la expresión de tabla común en la selección completa repetida, el tipo de datos, longitud y página de códigos para la columna se determinan basándose en la selección completa de inicialización. La columna correspondiente de la selección completa recursiva debe tener el mismo tipo de datos y longitud que el tipo de datos y longitud determinados en base a la selección completa de inicialización y la página de códigos debe coincidir (SQLSTATE 42825). Sin embargo, para los tipos de serie de caracteres, la longitud de los dos tipos de datos puede diferir. En este caso, la columna de la selección completa recursiva debe tener una longitud que podría asignarse siempre a la longitud determinada de la selección completa de inicialización.
- Cada selección completa que forma parte del ciclo de repetición no debe incluir ninguna función de columna, cláusula-group-by ni cláusula-having (SQLSTATE 42836).

  Las cláusulas FROM de estas selecciones completas pueden incluir como máximo una referencia a una expresión de tabla común que forme parte de un ciclo de repetición (SQLSTATE 42836).
- Ni la selección completa iterativa ni la selección completa recursiva global puede incluir una cláusula-order-by (SQLSTATE 42836).
- Las subconsultas (escalares o cuantificadas) no deben formar parte de ciclos de repetición (SQLSTATE 42836).

Cuando desarrolle expresiones de tabla comunes recursivas, recuerde que se puede crear un ciclo de repetición infinito (bucle). Compruebe que los ciclos de repetición terminen. Es muy importante si los datos implicados son cíclicos. Se espera que una expresión de tabla común recursiva incluya un predicado que impida un bucle infinito. Se espera que la expresión de tabla común recursiva incluya:

- Una selección completa recursiva, una columna de enteros incrementada por una constante.
- Un predicado en la cláusula where de la selección completa recursiva con el formato col_contador < constante" o "col_contador < :var_lengprinc".

Se emite un aviso si no se encuentra esta sintaxis en la expresión de tabla común recursiva (SQLSTATE 01605).

Ejemplo de recurrencia: Lista de material:

Las aplicaciones de tipo Lista de material (BOM) son una necesidad habitual en muchos entornos comerciales. Para ilustrar la capacidad de una expresión de tabla común recursiva para aplicaciones BOM, considere una tabla de piezas con subpiezas asociadas y la cantidad de subpiezas que se precisan en la pieza. Para este ejemplo, cree la tabla como se muestra a continuación:

```
CREATE TABLE PARTLIST
              (PIEZA VARCHAR(8),
               SUBPIEZA VARCHAR(8),
               CANTIDAD INTEGER);
```

Para obtener resultados de consulta en este ejemplo, supongamos que la tabla LISTA DE PIEZAS contiene los siguientes valores:

```
PIEZA    SUBPIEZA CANTIDAD
-------- -------- -----------
00       01                 5
00       05                 3
01       02                 2
01       03                 3
01       04                 4
01       06                 3
02       05                 7
02       06                 6
03       07                 6
04       08                10
04       09                11
05       10                10
05       11                10
06       12                10
06       13                10
07       14                 8
07       12                 8
```

Ejemplo 1: Explosión de primer nivel

El primer ejemplo se denomina explosión de primer nivel. Responde a la pregunta "Qué piezas son necesarias para crear la pieza identificada mediante '01'?". La lista incluirá las subpiezas directas, subpiezas de subpiezas, etc. Sin embargo, si una pieza se utiliza varias veces, las subpiezas correspondientes sólo aparecerán en la lista una vez.

```
WITH RPL (PIEZA, SUBPIEZA, CANTIDAD) AS
     (  SELECT PIEZA.RAIZ, SUBPIEZA.RAIZ, CANTIDAD.RAIZ
          FROM LISTA DE PIEZAS RAIZ
          WHERE PIEZA.RAIZ = '01'
       UNION ALL
        SELECT PIEZA.HIJA, SUBPIEZA.HIJA, CANTIDAD.HIJA
          FROM RPL PADRE, LISTA DE PIEZAS HIJA
          WHERE SUBPIEZA.PADRE = PIEZA.HIJA
     )
SELECT DISTINCT PIEZA, SUBPIEZA, CANTIDAD
 FROM RPL
  ORDER BY PIEZA, SUBPIEZA, CANTIDAD;
```

La consulta anterior incluye una expresión de tabla común, identificada mediante el nombre *RPL*, que expresa la pieza repetitiva de esta consulta. Ilustra los elementos básicos de una expresión de tabla común recursiva.

El primer operando (selección completa) de la UNION, al que se hace referencia como la *selección completa de inicialización*, obtiene los hijos directos de la pieza '01'. La cláusula FROM de esta selección completa hace referencia a la tabla fuente y nunca se hará referencia a sí misma (*RPL* en este caso). El resultado de la primera selección completa va a la expresión de tabla común *RPL* (LISTA DE PIEZAS recursiva). Como en este ejemplo, UNION debe ser siempre UNION ALL.

El segundo operando (selección completa) de UNION utiliza *RPL* para calcular las subpiezas de subpiezas haciendo que la cláusula FROM hará referencia a la expresión de tabla común *RPL* y la tabla fuente con una unión de una pieza de la tabla fuente (hija) a una subpieza del resultado actual contenido en *RPL* (padre). El resultado vuelve a *RPL* de nuevo. El segundo operando de UNION se utiliza entonces repetidamente hasta que ya no existan más hijas.

SELECT DISTINCT de la selección completa principal de esta consulta, garantiza que no aparezca en la lista la misma pieza/subpieza más de una vez.

El resultado de la consulta es el siguiente:

```
PIEZA    SUBPIEZA CANTIDAD
-------- -------- -----------
01       02                 2
01       03                 3
01       04                 4
01       06                 3
02       05                 7
02       06                 6
03       07                 6
04       08                10
04       09                11
05       10                10
05       11                10
06       12                10
06       13                10
07       12                 8
07       14                 8
```

Observe, en el resultado, que de la pieza '01' se pasa a la pieza '02', que a su vez pasa a la '06', etc. Observe también que la pieza '06' se alcanza dos veces, una a través de '01' directamente y otra a través de '02'. En el resultado, sin embargo, los subcomponentes sólo aparecen una vez en la lista (es el resultado de utilizar SELECT DISTINCT), tal como se requiere.

Es importante recordar que con las expresiones de tabla comunes recursivas puede generarse un *bucle infinito*. En este ejemplo, se produciría un bucle infinito si la condición de búsqueda del segundo operando que une las tablas madre e hija tuviera esta codificación:

```
SUBPIEZA.PADRE = SUBPIEZA.HIJA
```

Este ejemplo de bucle infinito es consecuencia de no codificar lo que se intenta codificar. Sin embargo, debe extremar la precaución al determinar qué es lo que se ha de codificar, de forma que se consiga un final definitivo del ciclo de recurrencia.

El resultado de esta consulta de ejemplo puede producirse en un programa de aplicación sin utilizar una expresión de tabla común recursiva. Sin embargo, ello requeriría iniciar una nueva consulta para cada nivel de repetición. Además, la aplicación necesita colocar de nuevo todos los resultados en la base de datos para ordenar el resultado. Todo ello hace que la lógica de la aplicación se complique y que el funcionamiento no sea el esperado. La lógica de la aplicación resulta aún más complicada e ineficaz para consultas de otras listas de material, tales como consultas resumidas y de explosión.

Ejemplo 2: Explosión resumida

El segundo ejemplo es una explosión resumida. La cuestión que se plantea aquí es la cantidad total de cada pieza que se requiere para crear la pieza '01'. La diferencia principal de la explosión de un solo nivel es la necesidad de agregar las cantidades. El primer ejemplo indica la cantidad de subpiezas necesarias para la pieza siempre que se requiera. No indica cuántas de las subpiezas se necesitan para crear la pieza '01'.

```
WITH RPL (PIEZA, SUBPIEZA, CANTIDAD) AS
   (
      SELECT PIEZA.RAIZ, SUBPIEZA.RAIZ, CANTIDAD.RAIZ
          FROM LISTA DE PIEZAS RAIZ
          WHERE PIEZA.RAIZ = '01'
       UNION ALL
      SELECT PIEZA.PADRE, SUBPIEZA.HIJA, CANTIDAD.PADRE*CANTIDAD.HIJA
```

```
         FROM RPL PADRE, LISTA DE PIEZAS HIJA
         WHERE SUBPIEZA.PADRE = PIEZA.HIJA
  )
SELECT PIEZA, SUBPIEZA, SUM(CANTIDAD) AS "CANT. total usada"
 FROM RPL
  GROUP BY PIEZA, SUBPIEZA
  ORDER BY PIEZA, SUBPIEZA;
```

En la consulta anterior, la lista de selección del segundo operando de UNION en la expresión de tabla común recursiva, identificada mediante el nombre *RPL*, muestra la agregación de la cantidad. Para averiguar qué porcentaje de subpieza se utiliza, la cantidad del elemento madre se multiplica por la cantidad por madre de una hija. Si una pieza se utiliza varias veces en lugares diferentes, requerirá otra agregación final. Esto se realiza por la agrupación por la expresión de tabla común *RPL* y utilizando la función de columna SUM en la lista de selección de la selección completa.

El resultado de la consulta es el siguiente:

```
PIEZA    SUBPIEZA Cant. total usada
-------- -------- --------------
01       02                    2
01       03                    3
01       04                    4
01       05                   14
01       06                   15
01       07                   18
01       08                   40
01       09                   44
01       10                  140
01       11                  140
01       12                  294
01       13                  150
01       14                  144
```

A la vista del resultado, considere la línea de la subpieza '06'. La cantidad total utilizada, con valor 15, deriva de la cantidad de 3 directamente para la pieza '01' y la cantidad de 6 para la pieza '02', que se necesita 2 veces en la pieza '01'.

Ejemplo 3: Control de profundidad

Puede surgir la cuestión de qué es lo que ocurre cuando existen más niveles de piezas en la tabla de los que está interesado para su consulta. Es decir, cómo se escribe una consulta para responder a la pregunta "Cuáles son los dos primeros niveles de piezas necesarias para crear la pieza identificada como '01'?" Por cuestiones de claridad en el ejemplo, el nivel se incluye en el resultado.

```
WITH RPL (NIVEL, PIEZA, SUBPIEZA, CANTIDAD) AS
      (
         SELECT 1,              PIEZA.RAIZ  SUBPIEZA.RAIZ, CANTIDAD.RAIZ
          FROM LISTA DE PIEZAS RAIZ
          WHERE PIEZA.RAIZ = '01'
       UNION ALL
         SELECT NIVEL+1.PADRE, PIEZA.HIJA, SUBPIEZA.HIJA, CANTIDAD.HIJA
          FROM RPL PADRE, LISTA DE PIEZAS HIJA
          WHERE SUBPIEZA.PADRE = PIEZA.HIJA
            AND NIVEL.PADRE < 2
      )
 SELECT PIEZA, NIVEL, SUBPIEZA, CANTIDAD
   FROM RPL;
```

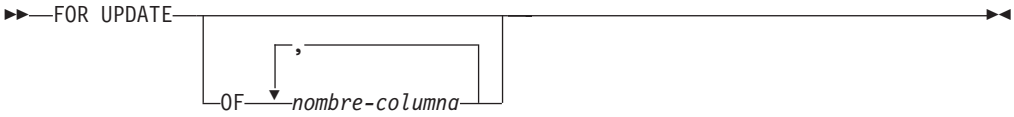
Esta consulta es similar al ejemplo 1. La columna *NIVEL* se ha introducido para contar los niveles desde la pieza original. En la selección completa de

inicialización, el valor de la columna *NIVEL* se inicializa en 1. En la selección completa subsiguiente, el nivel padre se incrementa en 1. A continuación, para controlar el número de niveles del resultado, la segunda selección completa incluye la condición de que el nivel padre debe ser menor que 2. Esto garantiza que la segunda selección completa sólo procesará hijos en el segundo nivel.

El resultado de la consulta es como sigue:

```
PIEZA    NIVEL       SUBPIEZA CANTIDAD
-------- ----------- -------- -----------
01                 1 02                 2
01                 1 03                 3
01                 1 04                 4
01                 1 06                 3
02                 2 05                 7
02                 2 06                 6
03                 2 07                 6
04                 2 08                10
04                 2 09                11
06                 2 12                10
06                 2 13                10
```

cláusula-update

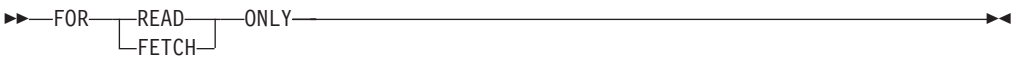


La cláusula FOR UPDATE identifica las columnas que se pueden actualizar en una sentencia UPDATE con posición posterior. Cada *nombre-columna* debe estar sin calificar y debe identificar una columna de la tabla o vista identificada en la primera cláusula FROM de la selección completa. Si la cláusula FOR UPDATE se especifica sin nombres de columna, se incluyen todas las columnas actualizables de la tabla o vista identificadas en la primera cláusula FROM de la selección completa.

La cláusula FOR UPDATE no puede utilizarse si es verdadera una de las siguientes situaciones:

- El cursor asociado con la sentencia select no se puede suprimir .
- Una de las columnas seleccionadas no es una columna actualizable de una tabla del catálogo y la cláusula FOR UPDATE no se ha utilizado para excluir dicha columna.

cláusula-read-only



La cláusula FOR READ ONLY indica que la tabla resultante es de sólo lectura y que, por lo tanto, no se puede hacer referencia al cursor en las sentencias UPDATE con posición y DELETE. FOR FETCH ONLY tiene el mismo significado.

Algunas tablas resultantes son de sólo lectura por naturaleza. (Por ejemplo, una tabla basada en una vista de sólo lectura.) Se puede seguir especificando FOR READ ONLY para dichas tablas, pero la especificación no surtirá efecto.

Para las tablas resultantes en las que están permitidas las actualizaciones y las supresiones, la especificación de FOR READ ONLY (o FOR FETCH ONLY) posiblemente mejorará el rendimiento de las operaciones FETCH, ya que permite al gestor de bases de datos realizar el bloqueo. Por ejemplo, en los programas que contienen sentencias de SQL dinámico sin la cláusula FOR READ ONLY ni ORDER BY, el gestor de bases de datos puede abrir cursores como si hubiese especificado la cláusula FOR UPDATE. Por lo tanto, se recomienda utilizar la cláusula FOR READ ONLY para mejorar el rendimiento, excepto en los casos en que se utilizarán las consultas en sentencias UPDATE o DELETE.

No se debe hacer referencia a una tabla resultante de sólo lectura en una sentencia UPDATE con posición o DELETE, ya sea de sólo lectura por naturaleza o especificada como FOR READ ONLY (FOR FETCH ONLY).

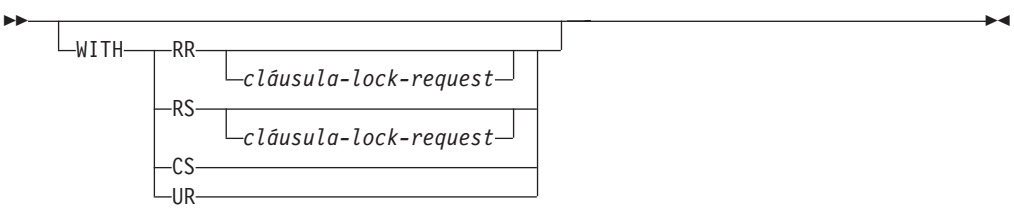
cláusula-optimize-for

La cláusula OPTIMIZE FOR pide el proceso especial de la *sentencia select*. Si se omite la cláusula, se supone que se recuperarán todas las filas de la tabla resultante; si se especifica, se supone que el número de filas recuperado probablemente no excederá de *n* donde *n* es el valor de *entero*. El valor de *n* debe ser un entero positivo. La utilización de la cláusula OPTIMIZE FOR influye en la optimización de la consulta basándose en la suposición de que se recuperarán *n* filas. Además, cuando los cursores están bloqueados, esta cláusula afecta al número de filas que se devuelven en cada bloque (es decir, no se devolverán más de *n* filas en cada bloque). Si se especifican la *cláusula-fetch-first* y la *cláusula-optimize-for*, se utilizará el valor entero menor de estas cláusulas para determinar el tamaño del almacenamiento intermedio de comunicaciones. Los valores se tienen en cuenta de forma independiente por motivos de optimización.

Esta cláusula no limita el número de filas que se pueden recuperar ni afecta al resultado de ninguna otra manera que no sea en el rendimiento. La utilización de OPTIMIZE FOR *n* ROWS puede mejorar el rendimiento si no se recuperan más de *n* filas, pero puede reducir el rendimiento si se recuperan más de *n* filas.

Si el valor de *n* multiplicado por el tamaño de la fila sobrepasa el tamaño del almacenamiento intermedio de comunicaciones, la cláusula OPTIMIZE FOR no tendrá ningún efecto sobre los almacenamientos intermedios de los datos. El tamaño del almacenamiento intermedio de comunicaciones está definido por el parámetro de configuración RQRIOBLK o ASLHEAPSZ.

cláusula-isolation



La *cláusula-isolation*, opcional, especifica el nivel de aislamiento en el que se ejecuta la sentencia y si debe obtenerse un tipo de bloqueo determinado.

- RR - Lectura repetible
- RS - Estabilidad de lectura
- CS - Estabilidad del cursor
- UR - Lectura no confirmada

El nivel de aislamiento por omisión de la sentencia es el nivel de aislamiento del paquete en el que está vinculada la sentencia.

cláusula-lock-request

La *cláusula-lock-request*, opcional, especifica el tipo de bloqueo que el gestor de bases de datos debe conseguir y retener:

SHARE Los procesos simultáneos pueden conseguir bloqueos SHARE o UPDATE sobre los datos.

UPDATE Los procesos simultáneos pueden conseguir bloqueos SHARE sobre los datos pero ningún proceso simultáneo puede conseguir un bloqueo UPDATE o EXCLUSIVE.

EXCLUSIVE Los procesos simultáneos no pueden conseguir un bloqueo sobre los datos.

La *cláusula-lock-request* se aplica a todas las exploraciones básicas de tabla y de índice que la consulta necesita, incluidas aquellas contenidas en subconsultas, funciones de SQL y métodos de SQL. No tiene ningún efecto sobre los bloqueos realizados por procedimientos, funciones externas o métodos externos. Todas las funciones de SQL o los métodos de SQL invocados (directa o indirectamente) por la sentencia deben crearse con INHERIT ISOLATION LEVEL WITH LOCK REQUEST (SQLSTATE 42601). La *cláusula-lock-request* no puede utilizarse con una consulta de modificación que pueda invocar a activadores o que necesite comprobaciones de la integridad referencial (SQLSTATE 42601).

Ejemplos de una sentencia-select

Ejemplo 1: Selección de todas las columnas y filas de la tabla EMPLOYEE.

```
SELECT * FROM EMPLOYEE
```

Ejemplo 2: Selección del nombre del proyecto (PROJNAME), la fecha de inicio (PRSTDATE) y la fecha de finalización (PRENDATE) de la tabla PROJECT. Ordenación de la tabla resultante por la fecha de finalización con las fechas más recientes primero.

```
SELECT PROJNAME, PRSTDATE, PRENDATE
 FROM PROJECT
 ORDER BY PRENDATE DESC
```

Ejemplo 3: Selección del número de departamento (WORKDEPT) y el salario medio del departamento (SALARY) para todos los departamentos de la tabla EMPLOYEE. Ordenación la tabla resultante por orden ascendente por el salario medio del departamento.

```
SELECT WORKDEPT, AVG(SALARY)
   FROM EMPLOYEE
   GROUP BY WORKDEPT
   ORDER BY 2
```

Ejemplo 4: Declaración de un cursor llamado UP_CUR para utilizarlo en un programa C para actualizar las columnas de fecha de inicio (PRSTDATE) y de fecha de finalización (PRENDATE) en la tabla PROJECT. El programa debe recibir los dos valores junto con el valor de número del proyecto (PROJNO) para cada fila.

```
EXEC SQL  DECLARE UP_CUR CURSOR FOR
              SELECT PROJNO, PRSTDATE, PRENDATE
                FROM PROJECT
                FOR UPDATE OF PRSTDATE, PRENDATE;
```

Ejemplo 5: Este ejemplo denomina a la expresión SAL+BONUS+COMM como TOTAL_PAY

```
  SELECT SALARY+BONUS+COMM AS TOTAL_PAY
    FROM EMPLOYEE
    ORDER BY TOTAL_PAY
```

Ejemplo 6: Determinación del número de empleado y el salario de los representantes de ventas junto con el salario medio y el número total de empleados de sus departamentos. También, listado del salario medio del departamento con el salario medio más alto.

La utilización de una expresión de tabla común para este caso ahorra la actividad de crear una vista DINFO como una vista normal. Durante la preparación de la sentencia, se evita el acceso al catálogo para la vista y, debido al contexto del resto de la selección completa, sólo se han de tener en cuenta las filas para el departamento de representantes de ventas para la vista.

```
WITH
   DINFO (DEPTNO, AVGSALARY, EMPCOUNT) AS
       (SELECT OTHERS.WORKDEPT, AVG(OTHERS.SALARY), COUNT(*)
         FROM EMPLOYEE OTHERS
         GROUP BY OTHERS.WORKDEPT
       ),
   DINFOMAX AS
       (SELECT MAX(AVGSALARY) AS AVGMAX FROM DINFO)
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY,
       DINFO.AVGSALARY, DINFO.EMPCOUNT, DINFOMAX.AVGMAX
 FROM EMPLOYEE THIS_EMP, DINFO, DINFOMAX
 WHERE THIS_EMP.JOB = 'SALESREP'
 AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

Ejemplo 7: Dadas dos tablas, EMPLOYEE y PROJECT, sustitución del empleado SALLY por el nuevo empleado GEORGE, asignación de todos los proyectos controlados por SALLY a GEORGE y devolución de los nombres de los proyectos actualizados.

```
WITH
  NEWEMP AS (SELECT EMPNO FROM NEW TABLE
              (INSERT INTO EMPLOYEE(EMPNO, FIRSTNME)
                  VALUES(NEXT VALUE FOR EMPNO_SEQ, 'GEORGE'))),
  OLDEMP AS (SELECT EMPNO FROM EMPLOYEE WHERE FIRSTNME = 'SALLY'),
  UPPROJ AS (SELECT PROJNAME FROM NEW TABLE
              (UPDATE PROJECT
                  SET RESPEMP = (SELECT EMPNO FROM NEWEMP)
                  WHERE RESPEMP = (SELECT EMPNO FROM OLDEMP))),
```

```
    DELEMP AS (SELECT EMPNO FROM OLD TABLE
                (DELETE FROM EMPLOYEE
                    WHERE EMPNO = (SELECT EMPNO FROM OLDEMP)))
  SELECT PROJNAME FROM UPPROJ;
```

Ejemplo 8: Recuperación de los datos de la tabla DEPT. Estos datos se actualizarán más tarde con una actualización buscada y deben estar bloqueados cuando se ejecute la consulta.

```
SELECT DEPTNO, DEPTNAME, MGRNO
  FROM DEPT
  WHERE ADMRDEPT ='A00'
  FOR READ ONLY WITH RS USE AND KEEP EXCLUSIVE LOCKS
```

Información relacionada:

- "Subselección" en la página 470
- "Sentencia DECLARE CURSOR" en la publicación *Consulta de SQL, Volumen 2*

Apéndice A. Límites de SQL

Las tablas siguientes describen algunos límites de SQL. Si el programador se ajusta al caso más restrictivo le servirá de ayuda para diseñar programas de aplicación que sean portátiles.

Tabla 28. Límites de longitud del identificador

Descripción	Límite en bytes
El nombre de autorización más largo (sólo puede ser de caracteres de un solo byte)	30
Nombre de restricción más largo	18
Nombre de correlación más largo	128
Nombre de condición más largo	64
Nombre de cursor más largo	18
Nombre (apodo) de columna no calificada más largo	128
Nombre más largo de índice de fuente de datos	128
Nombre de fuente de datos más largo	128
Nombre de tabla de fuente de datos más largo (*nombre-tabla-remota*)	128
Nombre de programa externo más largo	8
Identificador del lenguaje principal más largo [a]	255
Identificador de un usuario de fuente de datos más largo (*nombre-autorización-remota*)	30
Nombre de etiqueta más largo	64
Nombre de método más largo	18
Nombre de parámetro más largo [b]	128
Palabra clave más larga para acceder a una fuente de datos	32
Nombre de punto de salvar más largo	128
Nombre de esquema más largo [c]	30
Nombre de servidor (seudónimo de base de datos) más largo	8
Nombre de variable SQL más largo	64
Nombre de sentencia más largo	18
Nombre de grupo de transformación más largo	18
Nombre de columna no calificada más largo	30
Nombre de paquete no calificado más largo	8
Nombre más largo de tipo definido por el usuario no calificado, función definida por el usuario, método definido por el usuario, agrupación de almacenamientos intermedios, espacio de tablas, grupo de particiones de base de datos, activador, índice o especificación de índice	18
Nombre más largo de tabla no calificado, nombre de vista, procedimiento almacenado, apodo o seudónimo	128
Nombre más largo de reiniciador	128

Tabla 28. Límites de longitud del identificador (continuación)

Descripción	Límite en bytes

Notas:

1. [a] Los compiladores de lenguaje principal individuales pueden aplicar límites más restrictivos a los nombres de variables.
2. [b] En un procedimiento SQL la longitud del nombre de los parámetros está limitada a 64 bytes.
3. [c] El nombre de esquema para un tipo definido por el usuario está limitado a 8 bytes.

Tabla 29. Límites numéricos

Descripción	Límite
Valor INTEGER más pequeño	–2 147 483 648
Valor INTEGER máximo	+2 147 483 647
Valor BIGINT más pequeño	–9 223 372 036 854 775 808
Valor BIGINT máximo	+9 223 372 036 854 775 807
Valor SMALLINT mínimo	–32 768
Valor SMALLINT máximo	+32 767
Precisión decimal máxima	31
Valor DOUBLE mínimo	–1,79769E+308
Valor DOUBLE máximo	+1,79769E+308
Valor DOUBLE positivo mínimo	+2,225E–307
Valor DOUBLE negativo máximo	–2,225E–307
Valor REAL mínimo	–3,402E+38
Valor REAL máximo	+3,402E+38
Valor REAL positivo mínimo	+1,175E–37
Valor REAL negativo máximo	–1,175E–37

Tabla 30. Límites de series

Descripción	Límite
Longitud máxima de CHAR (en bytes)	254
Longitud máxima de VARCHAR (en bytes)	32 672
Longitud máxima de LONG VARCHAR (en bytes)	32 700
Longitud máxima de CLOB (en bytes)	2 147 483 647
Longitud máxima de GRAPHIC (en caracteres)	127
Longitud máxima de VARGRAPHIC (en caracteres)	16 336
Longitud máxima de LONG VARGRAPHIC (en caracteres)	16 350
Longitud máxima de DBCLOB (en caracteres)	1 073 741 823
Longitud máxima de BLOB (en bytes)	2 147 483 647
Longitud máxima de constante de caracteres	32 672
Longitud máxima de constante gráfica	16 336
Longitud máxima de series de caracteres concatenadas	2 147 483 647
Longitud máxima de series gráficas concatenadas	1 073 741 823
Longitud máxima de series binarias concatenadas	2 147 483 647

Tabla 30. Límites de series (continuación)

Descripción	Límite
El número máximo de dígitos de constante hexadecimal	16 336
Tamaño máximo de un comentario del catálogo (en bytes)	254
Instancia mayor de un objeto de una columna de tipo estructurado durante la ejecución (en gigabytes)	1

Tabla 31. Límites de fecha y hora

Descripción	Límite
Valor DATE mínimo	0001-01-01
Valor DATE máximo	9999-12-31
Valor TIME mínimo	00:00:00
Valor TIME máximo	24:00:00
Valor TIMESTAMP mínimo	0001-01-01-00.00.00.000000
Valor TIMESTAMP máximo	9999-12-31-24.00.00.000000

Tabla 32. Límites del gestor de bases de datos

Descripción	Límite
Máximo número de columnas en una tabla [g]	1 012
Máximo número de columnas en una vista [a]	5 000
Longitud máxima de una fila que incluye toda la actividad general [b g]	32 677
Tamaño máximo de una tabla por partición(en gigabytes) [c g]	512
Tamaño máximo de un índice por partición(en gigabytes)	512
El número máximo de filas en una tabla por partición	4 x 10^9
Clave de índice más larga que incluye toda la actividad general (en bytes)	1 024
El número máximo de columnas en una clave de índice	16
El número máximo de índices en una tabla	32 767 o almacenamiento
El número máximo de tablas de referencia en una sentencia de SQL o una vista	almacenamiento
El número máximo de declaraciones de variables del lenguaje principal en un programa precompilado[c]	almacenamiento
El número máximo de referencias de variables del lenguaje principal en una sentencia de SQL	32 767
Valor más largo de variable del lenguaje principal utilizado para operaciones de inserción o de actualización (en bytes)	2 147 483 647
Sentencia de SQL más larga (en bytes)	2 097 152
El número máximo de elementos en una lista de selección [g]	1 012
El número máximo de predicados en una cláusula WHERE o HAVING	almacenamiento
El número máximo de columnas en una cláusula GROUP BY [g]	1 012
Longitud total máxima de las columnas en una cláusula GROUP BY (en bytes)[g]	32 677

Tabla 32. Límites del gestor de bases de datos (continuación)

Descripción	Límite
El número máximo de columnas en una cláusula ORDER BY [g]	1 012
Longitud total máxima de las columnas en una cláusula ORDER BY (en bytes) [g]	32 677
Tamaño máximo de una SQLDA (en bytes)	almacenamiento
El número máximo de sentencias preparadas	almacenamiento
El número máximo de cursores declarados en un programa	almacenamiento
El número máximo de cursores abiertos a la vez	almacenamiento
El número máximo de tablas en un espacio de tablas SMS	65 534
El número máximo de restricciones en una tabla	almacenamiento
Nivel máximo de anidamiento de subconsultas	almacenamiento
El número máximo de subconsultas en una sola sentencia	almacenamiento
El número máximo de valores en una sentencia INSERT [g]	1 012
El número máximo de cláusulas SET en una sola sentencia UPDATE [g]	1 012
El número máximo de columnas en una restricción UNIQUE (soportado a través de un índice UNIQUE)	16
Longitud máxima combinada de las columnas de una restricción UNIQUE (soportada mediante un índice UNIQUE) (en bytes)	1 024
El número máximo de columnas de referencia en una clave foránea	16
Longitud máxima combinada de columnas de referencia en una clave foránea (en bytes)	1 024
Longitud máxima de una especificación de restricción de comprobación (en bytes)	65 535
El número máximo de columnas en una clave de particionamiento [e]	500
El número máximo de filas cambiadas en una unidad de trabajo	almacenamiento
El número máximo de paquetes	almacenamiento
El número máximo de constantes en una sentencia	almacenamiento
El número máximo de usuarios simultáneos del servidor [d]	64 000
El número máximo de parámetros en un procedimiento almacenado	32 767
El número máximo de parámetros en una función definida por el usuario	90
Profundidad máxima en tiempo de ejecución de activadores en cascada	16
El número máximo de supervisores de sucesos activos simultáneamente	32
Tamaño máximo de un espacio de tablas DMS normal (en gigabytes)[c] [g]	512
Tamaño máximo de un espacio de tablas DMS largo (en terabytes)[c]	2

Tabla 32. Límites del gestor de bases de datos (continuación)

Descripción	Límite
Tamaño máximo de un espacio de tablas DMS temporal (en terabytes)[c]	2
El número máximo de bases de datos por instancia en uso simultáneo	256
El número máximo de usuarios simultáneos por instancia	64 000
El número máximo de aplicaciones simultáneas por base de datos	60 000
El número máximo de conexiones por proceso dentro de un cliente DB2	512
Profundidad máxima de activadores en cascada	16
El número de partición máximo	999
El número máximo de objetos de tabla en un espacio de tablas DMS [f]	51 000
Parte más larga de la clave de índice variable (en bytes)[h]	1022 o almacenamiento
El número máximo de columnas en una vista o tabla de fuente de datos a la que se hace referencia mediante un apodo	5 000
Valor máximo de NPAGES en una agrupación de almacenamientos intermedios para emisiones de 32 bits	1 048 576
Valor máximo de NPAGES en una agrupación de almacenamientos intermedios para emisiones de 64 bits	2 147 483 647
Tamaño total máximo de todas las ranuras de almacenamientos intermedios de memoria (4K)	2 147 483 646
El número máximo de niveles de anidamiento para procedimientos almacenados	16
El número máximo de espacios de tablas en una base de datos	32 768
El número máximo de atributos en un tipo estructurado	4082
El número máximo de localizadores LOB abiertos simultáneamente en una transacción	32 100

Tabla 32. Límites del gestor de bases de datos (continuación)

Descripción	Límite

Notas:

1. [a] Este máximo puede conseguirse utilizando una unión en la sentencia CREATE VIEW. La selección de dicha vista está sujeta al límite del número máximo de elementos de una lista de selección.
2. [b] Los datos reales para las columnas BLOB, CLOB, LONG VARCHAR, DBCLOB y LONG VARGRAPHIC no se incluyen en esta cuenta. Sin embargo, la información acerca de la ubicación de los datos ocupa espacio en la fila.
3. [c] Los números mostrados son límites y aproximaciones arquitectónicos. En la práctica los límites pueden ser menores.
4. [d] El valor real será el valor del parámetro de configuración MAXAGENTS.
5. [e] Es un límite de arquitectura. El límite en la mayoría de las columnas de una clave de índice debe utilizarse como el límite práctico.
6. [f] Los objetos de tabla incluyen datos, índices, columnas LONG VARCHAR o VARGRAPHIC y columnas LOB. Los objetos de tabla que están en el mismo espacio de tablas que los datos de tabla no cuentan como adicionales respecto al límite. No obstante, cada objeto de tabla que está en un espacio de tablas diferente de los datos de tabla representa uno respecto al límite para cada tipo de objeto de tabla por tabla en el espacio de tablas en que reside el objeto de tabla.
7. [g] Para ver los valores relativos al tamaño, vea la tabla Tabla 33.
8. [h] Está limitado solamente por la clave de índice más larga que incluye toda la actividad general (en bytes). A medida que aumenta el número de partes de claves de índice, disminuye la longitud máxima de cada parte de clave.

Tabla 33. Límites específicos de tamaño de página del gestor de bases de datos

Descripción	Límite de tamaño de página de 4K	Límite de tamaño de página de 8K	Límite de tamaño de página de 16K	Límite de tamaño de página de 32K
Máximo número de columnas en una tabla	500	1 012	1 012	1 012
Longitud máxima de una fila que incluye toda la actividad general	4 005	8 101	16 293	32 677
Tamaño máximo de una tabla por partición (en gigabytes)	64	128	256	512
Tamaño máximo de un índice por partición(en gigabytes)	64	128	256	512
El número máximo de elementos en una lista de selección	500	1 012	1 012	1 012
El número máximo de columnas en una cláusula GROUP BY	500	1 012	1 012	1 012
Longitud total máxima de las columnas en una cláusula GROUP BY (en bytes)	4 005	8 101	16 293	32 677
El número máximo de columnas en una cláusula ORDER BY	500	1 012	1 012	1 012
Longitud total máxima de las columnas en una cláusula ORDER BY (en bytes)	4 005	8 101	16 293	32 677
El número máximo de valores en una sentencia INSERT	500	1 012	1 012	1 012

Tabla 33. Límites específicos de tamaño de página del gestor de bases de datos (continuación)

Descripción	Límite de tamaño de página de 4K	Límite de tamaño de página de 8K	Límite de tamaño de página de 16K	Límite de tamaño de página de 32K
El número máximo de cláusulas SET en una sola sentencia UPDATE	500	1 012	1 012	1 012
Tamaño máximo de un espacio de tablas DMS normal (en gigabytes)	64	128	256	512

Información relacionada:

- "maxagents - Maximum number of agents configuration parameter" en la publicación *Administration Guide: Performance*

Apéndice B. SQLCA (área de comunicaciones SQL)

Una SQLCA es un conjunto de variables que se actualiza al final de la ejecución de cada sentencia de SQL. Un programa que contiene sentencias de SQL ejecutables y se precompila con la opción LANGLEVEL SAA1 (el valor por omisión) o MIA debe proporcionar exactamente una SQLCA, aunque es posible que exista más de una SQLCA por paso en una aplicación de múltiples pasos.

Cuando se precompila un programa con la opción LANGLEVEL SQL92E, puede declararse una variable SQLCODE o SQLSTATE en la sección de declaración SQL o se puede declarar una variable SQLCODE en algún otro lugar del programa.

No se debe proporcionar ninguna SQLCA cuando se utiliza LANGLEVEL SQL92E. La sentencia de SQL INCLUDE puede utilizarse para proporcionar la declaración de la SQLCA en todos los lenguajes excepto en REXX. La SQLCA se proporciona automáticamente en REXX.

Para visualizar la SQLCA después de cada mandato ejecutado a través del procesador de línea de mandatos, utilice el mandato **db2 -a**. La SQLCA se proporciona como parte de la salida para los mandatos posteriores. La SQLCA también se vuelca en el archivo db2diag.log.

Descripción de los campos de la SQLCA

Tabla 34. Campos de la SQLCA. Los nombres de los campos que se muestran son aquellos presentes en una SQLCA obtenida mediante una sentencia INCLUDE.

Nombre	Tipo de datos	Valores de campos
sqlcaid	CHAR(8)	Una indicación visual para los vuelcos de almacenamiento que contienen la 'SQLCA'. El sexto byte es 'L' si el número de línea devuelto procede del análisis sintáctico del cuerpo de un procedimiento SQL.
sqlcabc	INTEGER	Contiene la longitud de la SQLCA, 136.
sqlcode	INTEGER	Contiene el código de retorno SQL. **Código** **Significado** **0** Ejecución satisfactoria (aunque pueden haberse establecido uno o varios indicadores SQLWARN). **positivo** Ejecución satisfactoria, pero con una condición de aviso. **negativo** Condición de error.
sqlerrml	SMALLINT	Indicador de longitud para *sqlerrmc*, en el rango de 0 a 70.0 significa que el valor de *sqlerrmc* no es relevante.

Tabla 34. Campos de la SQLCA (continuación). Los nombres de los campos que se muestran son aquellos presentes en una SQLCA obtenida mediante una sentencia INCLUDE.

Nombre	Tipo de datos	Valores de campos
sqlerrmc	VARCHAR (70)	Contiene uno o más símbolos, separados por X'FF', que sustituyen a las variables en las descripciones de las condiciones de error. Este campo también se utiliza cuando se establece una conexión satisfactoria. Cuando se emite una sentencia de SQL compuesta NOT ATOMIC, puede contener información acerca de un máximo de siete errores. El último símbolo puede ir seguido de X'FF'. El valor *sqlerrml* incluirá los X'FF' de cola existentes.
sqlerrp	CHAR(8)	Empieza con un identificador de tres letras que indica el producto, seguido de cinco dígitos indicando la versión, release y nivel de modificación del producto. Por ejemplo, SQL08010 significa DB2 Universal Database Versión 8, Release 1, Nivel de modificación 0. Si SQLCODE indica una condición de error, este campo identifica el módulo que ha devuelto el error. Este campo también se utiliza cuando se establece una conexión satisfactoria.
sqlerrd	ARRAY	Seis variables INTEGER que proporcionan información de diagnóstico. Generalmente, estos valores están vacíos si no hay errores, excepto sqlerrd(6) de una base de datos particionada.
sqlerrd(1)	INTEGER	Si se invoca la conexión y es satisfactoria, contiene la diferencia máxima esperada en la longitud de los datos de caracteres mixtos (tipos de datos CHAR) cuando se convierten a la página de códigos de la base de datos de la página de códigos de la aplicación. Un valor de 0 ó 1 indica sin expansión; un valor mayor que 1 indica una posible expansión en longitud; un valor negativo indica una posible contracción. Cuando un procedimiento SQL termina satisfactoriamente su ejecución, este campo contiene el valor del estado de retorno del procedimiento SQL.
sqlerrd(2)	INTEGER	Si se invoca la conexión y es satisfactoria, contiene la diferencia máxima esperada en la longitud de datos de caracteres mixtos (tipos de datos CHAR) cuando se convierten a la página de códigos de la aplicación de la página de códigos de la base de datos. Un valor de 0 ó 1 indica sin expansión; un valor mayor que 1 indica una posible expansión en longitud; un valor negativo indica una posible contracción. Si la SQLCA es el resultado de una sentencia de SQL compuesta NOT ATOMIC que ha encontrado uno o varios errores, el valor se establece en el número de sentencias que han fallado.

Tabla 34. Campos de la SQLCA (continuación). Los nombres de los campos que se muestran son aquellos presentes en una SQLCA obtenida mediante una sentencia INCLUDE.

Nombre	Tipo de datos	Valores de campos
sqlerrd(3)	INTEGER	Si se invoca PREPARE y la operación es satisfactoria, contiene una estimación del número de filas que se devolverán. Después de INSERT, UPDATE, DELETE o MERGE, contiene el número real de filas que estaban cualificadas para la operación. Si se invoca SQL compuesto, contiene una acumulación de todas las filas de subsentencia. Si se invoca CONNECT, contiene 1 si la base de datos se puede actualizar, o 2 si la base de datos es de sólo lectura. Si se invoca la sentencia OPEN y el cursor contiene sentencias de cambio de datos de SQL, este campo contiene la suma del número de filas cualificadas para las operaciones incorporadas de inserción, actualización, supresión o fusión. Si se invoca CREATE PROCEDURE para un procedimiento SQL y se detecta un error durante el análisis del cuerpo del procedimiento SQL, contiene el número de línea donde se encontró el error. El sexto byte de sqlcaid debe ser 'L' para que este valor sea un número de línea válido.
sqlerrd(4)	INTEGER	Si se invoca PREPARE y es satisfactoria, contiene una estimación del coste relativo de los recursos necesarios para procesar la sentencia. Si se invoca SQL compuesto, contiene una cuenta del número de subsentencias satisfactorias. Si se invoca CONNECT, contiene 0 para una confirmación en una fase de un cliente de nivel inferior; 1 para una confirmación de una fase; 2 para confirmación de sólo lectura de una fase; y 3 para una confirmación de dos fases.
sqlerrd(5)	INTEGER	Contiene el número total de filas suprimidas, insertadas o actualizadas como resultado de: • La imposición de las restricciones después de una operación de supresión satisfactoria • El proceso de sentencias de SQL activadas por activadores activados. Si se invoca SQL compuesto, contiene una acumulación del número de dichas filas para todas las subsentencias. En algunos casos cuando se encuentra un error, este campo contiene un valor negativo que es un puntero de un error interno. Si se invoca CONNECT, contiene el valor de tipo de autentificación 0 para una autentificación de servidor; 1 para la autentificación de cliente; 2 para la autentificación utilizando DB2 Connect; 4 para autentificación SERVER_ENCRYPT; 5 para autentificación utilizando DB2 Connect con cifrado; 7 para autentificación KERBEROS; 8 para autentificación KRB_SERVER_ENCRYPT; 9 para autentificación GSSPLUGIN; 10 para autentificación GSS_SERVER_ENCRYPT y 255 para una autentificación sin especificar.

Tabla 34. Campos de la SQLCA (continuación). Los nombres de los campos que se muestran son aquellos presentes en una SQLCA obtenida mediante una sentencia INCLUDE.

Nombre	Tipo de datos	Valores de campos
sqlerrd(6)	INTEGER	Para una base de datos particionada, contiene el número de partición de la partición que ha encontrado el error o aviso. Si no se han encontrado errores ni avisos, este campo contiene el número de partición del nodo coordinador. El número de este campo es igual al especificado para la partición del archivo db2nodes.cfg.
sqlwarn	Matriz	Un conjunto de indicadores de aviso, que contiene cada uno un blanco o W. Si se invoca SQL compuesto, contiene una acumulación de indicadores de aviso establecidos para todas las subsentencias.
sqlwarn0	CHAR(1)	Blanco si todos los demás indicadores están en blanco; contiene W si como mínimo otro indicador no está en blanco.
sqlwarn1	CHAR(1)	Contiene una ″W″ si el valor de una columna de serie se ha truncado cuando se ha asignado a una variable del lenguaje principal. Contiene una ″N″ si el terminador nulo se ha truncado.Contiene A si la operación CONNECT o ATTACH se realiza satisfactoriamente y el nombre de autorización de la conexión tiene más de 8 bytes. Contiene P si el coste estimado relativo de la sentencia PREPARE desqlerrd(4) sobrepasaba el valor que podía almacenarse en un INTEGER o era menor que 1 y el registro especial CURRENT EXPLAIN MODE o CURRENT EXPLAIN SNAPSHOT se ha definido en un valor distinto de NO.
sqlwarn2	CHAR(1)	Contiene W si los valores nulos se han eliminado del argumento de una función. [a]
sqlwarn3	CHAR(1)	Contiene W si el número de columnas no es igual al número de variables del lenguaje principal. Contiene Z si el número de localizadores del conjunto de resultados especificado en la sentencia ASSOCIATE LOCATORS es menor que el número de conjuntos de resultados devuelto por un procedimiento.
sqlwarn4	CHAR(1)	Contiene W si una sentencia UPDATE o DELETE preparada no incluye una cláusula WHERE.
sqlwarn5	CHAR(1)	Reservado para su utilización en el futuro.
sqlwarn6	CHAR(1)	Contiene W si se ha ajustado el resultado del cálculo de una fecha para evitar una fecha imposible.
sqlwarn7	CHAR(1)	Reservado para su utilización en el futuro. Si se invoca CONNECT y se ejecuta satisfactoriamente, contiene una ″E″ si el parámetro de configuración DYN_QUERY_MGMT de la base de datos está habilitado.
sqlwarn8	CHAR(1)	Contiene W si el carácter que no se ha podido convertir se ha sustituido por un carácter de sustitución.
sqlwarn9	CHAR(1)	Contiene W si se han pasado por alto las expresiones aritméticas que contienen errores durante el proceso de función de columna.
sqlwarn10	CHAR(1)	Contiene W si ha habido un error de conversión al convertir un valor de datos de caracteres de uno de los campos de la SQLCA.

Tabla 34. Campos de la SQLCA (continuación). Los nombres de los campos que se muestran son aquellos presentes en una SQLCA obtenida mediante una sentencia INCLUDE.

Nombre	Tipo de datos	Valores de campos
sqlstate	CHAR(5)	Un código de retorno que indica el resultado de la sentencia de SQL ejecutada más recientemente.

[a] Es posible que algunas funciones no establezcan SQLWARN2 en W aunque se hayan eliminado los valores nulos, porque el resultado no dependía de la eliminación de dichos valores.

Informe de errores

El orden en que se informa de los errores es el siguiente:

1. Las condiciones de error graves siempre se informan. Cuando se informa de un error grave, no hay ninguna adición a la SQLCA.
2. Si no se produce ningún error grave, un error de punto muerto tiene prioridad sobre los demás errores.
3. En el resto de errores, se devuelve la SQLCA para el primer código SQL negativo.
4. Si no se detecta ningún código SQL negativo, se devuelve la SQLCA para el primer aviso (es decir, código SQL positivo).

   En un sistema de base de datos particionada, se produce una excepción a esta regla si se invoca una operación de manipulación de datos en una tabla que está vacía en una partición, pero que tiene datos en otras particiones. Sólo se devuelve SQLCODE +100 a la aplicación si los agentes de todas las particiones devuelven SQL0100W, porque la tabla está vacía en todas las particiones o porque no hay más filas que cumplan la cláusula WHERE de una sentencia UPDATE.

Utilización de SQLCA en sistemas de bases de datos particionadas

En sistemas de bases de datos particionadas, una sentencia de SQL pueden ejecutarla varios agentes de distintas particiones y cada agente puede devolver una SQLCA distinta para diferentes errores o avisos. El agente coordinador también tiene su propia SQLCA.

Para proporcionar una vista coherente para las aplicaciones, todos los valores de SQLCA se fusionan en una estructura y los campos de SQLCA indican cuentas globales como, por ejemplo:

- Para todos los errores y avisos, el campo *sqlwarn* contiene los distintivos de aviso recibidos de todos los agentes.
- Los valores de los campos *sqlerrd* que indican cuentas de fila son acumulaciones de todos los agentes.

Observe que es posible que no se devuelva SQLSTATE 09000 cada vez que se produzca un error durante el proceso de una sentencia de SQL activada.

Apéndice C. SQLDA (área de descriptores de SQL)

Una SQLDA es un conjunto de variables que son necesarias para la ejecución de la sentencia de SQL DESCRIBE. Las variables SQLDA son opciones que las sentencias PREPARE, OPEN, FETCH y EXECUTE pueden utilizar. Una SQLDA se comunica con SQL dinámico; puede utilizarse en una sentencia DESCRIBE, modificarse con las direcciones de las variables del lenguaje principal y después volverse a utilizar en una sentencia FETCH o EXECUTE.

Se da soporte a SQLDA para todos los lenguajes, pero sólo se proporcionan declaraciones predefinidas para C, REXX, FORTRAN y COBOL.

El significado de la información de una SQLDA depende de su utilización. En PREPARE y DESCRIBE, una SQLDA proporciona información para un programa de aplicación acerca de una sentencia preparada. En OPEN, EXECUTE y FETCH, una SQLDA describe las variables del lenguaje principal.

En DESCRIBE y PREPARE, si cualquiera de las columnas que se describen es de tipo LOB (los localizadores de LOB y las variables de referencia de archivo no necesitan doblar las SQLDA), de tipo de referencia o un tipo definido por el usuario, el número de entradas de SQLVAR para la SQLDA completa se doblará. Por ejemplo:

- Cuando se describe una tabla con 3 columnas VARCHAR y 1 columna INTEGER, habrán 4 entradas SQLVAR
- Cuando se describe una tabla con 2 columnas VARCHAR, 1 columna CLOB y 1 columna de enteros, habrán 8 entradas SQLVAR

En EXECUTE, FETCH y OPEN, si cualquiera de las variables que se describen es de tipo LOB (los localizadores de LOB y las variables de referencia de archivos no necesitan doblar las SQLDA), de tipo estructurado, el número de entradas de SQLVAR para la SQLDA completa debe doblarse. (Estos casos no son aplicables a los tipos diferenciados ni a los tipos de referencia, porque la base de datos no necesita la información adicional que proporcionan las entradas dobles).

Descripción de los campos de la SQLDA

Una SQLDA consta de cuatro variables seguidas de un número arbitrario de ocurrencias de una secuencia de variables llamadas en conjunto SQLVAR. En OPEN, FETCH y EXECUTE, cada ocurrencia de SQLVAR describe una variable del lenguaje principal. En DESCRIBE y PREPARE, cada ocurrencia de SQLVAR describe una columna de una tabla resultante o un marcador de parámetros. Hay dos tipos de entradas SQLVAR:

- **SQLVAR base:** Estas entradas siempre están presentes. Contienen la información base acerca de la columna, el marcador de parámetros o la variable del lenguaje principal como, por ejemplo, el código del tipo de datos, el atributo de longitud, el nombre de la columna, la dirección de la variable del lenguaje principal y la dirección de la variable indicadora.
- **SQLVAR secundarias:** Estas entradas sólo están presentes si el número de entradas SQLVAR se dobla por las reglas indicadas arriba. Para los tipos definidos por el usuario (diferenciado o estructurado), contienen el nombre del tipo definido por el usuario. Para los tipos de referencia, contienen el tipo de destino de la referencia. Para los LOB, contienen el atributo de longitud de la

variable del lenguaje principal y un puntero que indica el almacenamiento intermedio que contiene la longitud real. (La información de tipo diferenciado y de LOB no se solapa, por lo que los tipos diferenciados pueden estar basados en LOB sin hacer que se triplique el número de entradas de SQLVAR en DESCRIBE). Si se utilizan localizadores o variables de referencia a archivos para representar los LOB, estas entradas no son necesarias.

En las SQLDA que contienen ambos tipos de entradas, las SQLVAR base están en un bloque antes del bloque de SQLVAR secundarias. En cada una, el número de entradas es igual al valor de SQLD (incluso aunque muchas de las entradas SQLVAR secundarias pueden estar sin utilizar).

Las circunstancias bajo las que DESCRIBE establece las entradas SQLVAR se detallan en el apartado “Efecto de DESCRIBE en la SQLDA” en la página 544.

Campos en la cabecera SQLDA

Tabla 35. Campos en la cabecera SQLDA

Nombre C	Tipo de datos SQL	Uso en DESCRIBE y PREPARE (establecido por el gestor de bases de datos excepto para SQLN)	Uso en FETCH, OPEN y EXECUTE (establecido por la aplicación antes de ejecutar la sentencia)
sqldaid	CHAR(8)	El séptimo byte de este campo es un byte de distintivo llamado SQLDOUBLED. El gestor de bases de datos establece SQLDOUBLED en el carácter ’2’ si se han creado dos entradas SQLVAR para cada columna; de lo contrario, se establece en un blanco (X'20' en ASCII, X'40' en EBCDIC). Consulte el apartado “Efecto de DESCRIBE en la SQLDA” en la página 544 para ver los detalles de cuándo se establece SQLDOUBLED.	El séptimo byte de este campo se utiliza cuando se dobla el número de SQLVAR. Se denomina SQLDOUBLED. Si alguna de las variables del lenguaje principal que se describen es un tipo estructurado, BLOB, CLOB o DBCLOB, el séptimo byte debe establecerse en el carácter ’2’; en otro caso puede establecerse en cualquier carácter, pero es aconsejable utilizar un espacio en blanco.
sqldabc	INTEGER	Para 32 bits, la longitud de la SQLDA, que es igual a SQLN*44+16. Para 64 bits, la longitud de la SQLDA, que es igual a SQLN*56+16	Para 32 bits, la longitud de la SQLDA, que es >= SQLN*44+16. Para 64 bits, la longitud de la SQLDA, que es >= SQLN*56+16.
sqln	SMALLINT	Sin cambiar por el gestor de bases de datos. Debe estar establecido en un valor mayor o igual que cero antes de que se ejecute la sentencia DESCRIBE. Indica el número total de ocurrencias de SQLVAR.	El número total de ocurrencias de SQLVAR proporcionadas en la SQLDA. SQLN debe estar establecido en un valor mayor o igual que cero.
sqld	SMALLINT	Establecido por el gestor de bases de datos en el número de columnas de la tabla resultante o en el número de marcadores de parámetros.	El número de variables del lenguaje principal descritas por las ocurrencias de SQLVAR.

Campos de una ocurrencia de una SQLVAR base

Tabla 36. Campos en una SQLVAR base

Nombre	Tipo de datos	Uso en DESCRIBE y PREPARE	Uso en FETCH, OPEN y EXECUTE
sqltype	SMALLINT	Indica el tipo de datos de la columna o el marcador de parámetros y si puede contener nulos. (Los marcadores de parámetros siempre se consideran anulables.) Tabla 38 en la página 545 lista todos los valores permitidos y sus significados. Observe que para un tipo de referencia diferenciado, el tipo de datos del tipo base se coloca en este campo. Para un tipo estructurado, el tipo de datos del resultado de la función de transformación FROM SQL del grupo de transformación (basado en el registro especial CURRENT DEFAULT TRANSFORM GROUP) se coloca en este campo. No existe ninguna indicación en la SQLVAR base de que forma parte de la descripción de un tipo definido por el usuario o tipo de referencia.	Igual que para la variable del lenguaje principal. Las variables del lenguaje principal para los valores de fecha y hora deben ser variables de serie de caracteres. Para FETCH, un código de tipo de fecha y hora significa una serie de caracteres de longitud fija. Si sqltype es un valor de número par, se ignora el campo sqlind.
sqllen	SMALLINT	El atributo de longitud de la columna o el marcador de parámetros. Para columnas y marcadores de parámetros de fecha y hora, la longitud de la representación de serie de los valores. Véase Tabla 38 en la página 545. Observe que el valor está establecido en 0 para las series de objeto grande (incluso para aquellas cuyo atributo de longitud sea lo suficientemente pequeño como para caber en un entero de dos bytes).	El atributo de longitud de la variable del lenguaje principal. Véase Tabla 38 en la página 545. Observe que el gestor de bases de datos ignora el valor para las columnas CLOB, DBCLOB y BLOB. Se utiliza el campo len.sqllonglen de la SQLVAR secundaria en su lugar.
sqldata	puntero	En las SQLVAR de series, sqldata contiene la página de códigos. En las SQLVAR de serie-caracteres, si la columna está definida con el atributo FOR BIT DATA, sqldata contiene 0. En otras SQLVAR de series-caracteres, sqldata contiene la página de códigos SBCS, para datos SBCS, o la página de códigos SBCS asociada a la página de códigos MBCS compuesta, para datos MBCS. Para las SQLVAR de series de caracteres de japonés EUC, chino tradicional EUC e Unicode UTF-8,sqldata contiene 954, 964 y 1208, respectivamente. Para todos los tipos de columna, sqldata es indefinido.	Contiene la dirección de la variable del lenguaje principal (donde se almacenarán los datos leídos).

Campos de una ocurrencia de una SQLVAR base

Tabla 36. Campos en una SQLVAR base (continuación)

Nombre	Tipo de datos	Uso en DESCRIBE y PREPARE	Uso en FETCH, OPEN y EXECUTE
sqlind	puntero	En las SQLVAR de series-caracteres, sqlind contiene 0, excepto para los datos MBCS, si sqlind contiene la página de códigos DBCS asociada con la página de códigos MBCS compuesta. Para el resto de tipos, sqlind no está definido.	Contiene la dirección de una variable indicadora asociada si existe una; de lo contrario, no se utiliza. Si sqltype es un valor de número par, se ignora el campo sqlind.
sqlname	VARCHAR (30)	Contiene el nombre no calificado de la columna o el marcador de parámetros. Para columnas y marcadores de parámetros que tengan el nombre generado por el sistema, el byte número trece se establece en X'FF'. Para los nombres de columna especificados por la cláusula AS, el byte es X'00'.	Cuando se utiliza DB2 Connect para acceder al servidor, se puede establecer sqlname para que indique una serie FOR BIT DATA de la forma siguiente: • la longitud de sqlname es 8 • los cuatro primeros bytes de sqlname son X'00000000' • los cuatro bytes restantes de sqlname están reservados (y se ignoran actualmente).

Campos de una ocurrencia de una SQLVAR secundaria

Tabla 37. Campos en una SQLVAR secundaria

Nombre	Tipo de datos	Uso en DESCRIBE y PREPARE	Uso en FETCH, OPEN y EXECUTE
len.sqllonglen	INTEGER	El atributo de longitud de una columna o marcador de parámetros de BLOB, CLOB o DBCLOB.	El atributo de longitud de una variable del lenguaje principal BLOB, CLOB o DBCLOB. El gestor de bases de datos pasa por alto el campo SQLLEN de la SQLVAR base para los tipos de datos. El atributo de longitud almacena el número de bytes para BLOB o CLOB y el número de caracteres para DBCLOB.
reservado2	CHAR(3) para 32 bits, y CHAR(11) para 64 bits.	No utilizado.	No utilizado.
sqlflag4	CHAR(1)	El valor es X'01' si la SQLVAR representa un tipo de referencia con un tipo de destino denominado en sqldatatype_name. El valor es X'12' si SQLVAR representa un tipo estructurado, y nombre_tipodatosSQL contiene el nombre del tipo definido por el usuario. En otro caso, el valor es X'00'.	Se establece en X'01' si la SQLVAR representa un tipo de referencia con un tipo de destino mencionado en nombre_tipodatosSQL. El valor es X'12' si SQLVAR representa un tipo estructurado, y nombre_tipodatosSQL contiene el nombre del tipo definido por el usuario. En otro caso, el valor es X'00'.

Tabla 37. Campos en una SQLVAR secundaria (continuación)

Nombre	Tipo de datos	Uso en DESCRIBE y PREPARE	Uso en FETCH, OPEN y EXECUTE
sqldatalen	puntero	No utilizado.	Utilizado solamente para las variables del lenguaje principal BLOB, CLOB y DBCLOB. Si este campo es NULL, entonces la longitud real (en caracteres) debe almacenarse en los 4 bytes inmediatamente antes del inicio de los datos y SQLDATA debe apuntar al primer byte de la longitud de campo. Si este campo no es NULL, contiene un puntero que indica un almacenamiento intermedio de 4 bytes de longitud que contiene la longitud real *en bytes* (incluso para DBCLOB) de los datos del almacenamiento intermedio al que apunta el campo de SQLDATA de la SQLVAR base coincidente. Observe que, sin tener en cuenta si este campo se utiliza o no, debe establecerse el campo len.sqllonglen.
nombre_sqldatatype	VARCHAR(27)	Para un tipo definido por el usuario, el gestor de bases de datos establece este campo en el nombre del tipo definido por el usuario, calificado al completo.[1] Para un tipo de referencia, el gestor de bases de datos establece este campo en el nombre del tipo calificado al completo del tipo de destino de la referencia.	Para los tipos estructurados, se establece en el nombre del tipo definido por el usuario, calificado al completo, con el formato indicado en la nota de la tabla.[1]
reservado	CHAR(3)	No utilizado.	No utilizado.

[1] Los 8 primeros bytes contienen el nombre de esquema del tipo (ampliado por la derecha con espacios, si es necesario). El byte 9 contiene un punto (.). Los bytes del 10 al 27 contienen la parte de orden inferior del nombre de tipo, que *no* se extiende por la derecha con espacios.

Tenga en cuenta que, aunque el principal propósito de este campo es para el nombre de los tipos definidos por el usuario, el campo también se establece para los tipos de datos predefinidos por IBM. En este caso, el nombre de esquema es SYSIBM y la parte de orden inferior del nombre es el nombre almacenado en la columna TYPENAME de la vista de catálogo DATATYPES. Por ejemplo:

```
Nombre tipo      longitud nombre_sqldatatype
---------        ------   ----------------
A.B              10       A       .B
INTEGER          16       SYSIBM  .INTEGER
"Frank's".SMINT  13       Frank's .SMINT
MY."type  "      15       MY      .type
```

Efecto de DESCRIBE en la SQLDA

Para una sentencia DESCRIBE OUTPUT o PREPARE OUTPUT INTO, el gestor de bases de datos siempre establece SQLD en el número de columnas del conjunto resultante o en el número de marcadores de parámetro de la salida. Para una sentencia DESCRIBE INPUT o PREPARE INPUT INTO, el gestor de bases de datos siempre establece SQLD en el número de marcadores de parámetros de entrada de la sentencia. Observe que un marcador de parámetros que se corresponda con un parámetro INOUT en una sentencia CALL se describe en los descriptores tanto de entrada como de salida.

Las SQLVAR de la SQLDA se establecen en los casos siguientes:

- SQLN >= SQLD y ninguna entrada es un LOB, un tipo definido por el usuario o un tipo de referencia

  Se establecen las primeras entradas SQLD SQLVAR y SQLDOUBLED se establece en blanco.
- SQLN >= 2*SQLD y como mínimo una entrada es un LOB, un tipo definido por el usuario o un tipo de referencia

  Las entradas SQLD SQLVAR se establecen dos veces y SQLDOUBLED se establece en '2'.
- SQLD <= SQLN < 2*SQLD y como mínimo una entrada es un tipo diferenciado o un tipo de referencia, pero no hay entradas LOB ni entradas de tipo estructurado

  Se establecen las primeras entradas SQLD SQLVAR y SQLDOUBLED se establece en blanco. Si la opción SQLWARN es YES, se emite un aviso SQLCODE +237 (SQLSTATE 01594).

Las SQLVAR de la SQLDA NO se establecen (es necesaria la asignación de espacio adicional y otra DESCRIBE) en los casos siguientes:

- SQLN < SQLD y ninguna entrada es un LOB, un tipo definido por el usuario o un tipo de referencia

  No se establece ninguna entrada SQLVAR y SQLDOUBLED se establece en blanco. Si la opción SQLWARN es YES, se emite un aviso SQLCODE +236 (SQLSTATE 01005).

  Asigne las SQLD SQLVAR para una operación DESCRIBE satisfactoria.
- SQLN < SQLD y como mínimo una entrada es un tipo diferenciado o un tipo de referencia, pero no hay entradas LOB ni entradas de tipo estructurado

  No se establece ninguna entrada SQLVAR y SQLDOUBLED se establece en blanco. Si la opción SQLWARN es YES, se emite un aviso SQLCODE +239 (SQLSTATE 01005).

  Asigne un número de estructuras SQLVAR igual a 2*SQLD para lograr una operación DESCRIBE satisfactoria que incluya los nombres de los tipos diferenciados y tipos destino de tipos de referencia.
- SQLN < 2*SQLD y como mínimo una entrada es un LOB o un tipo estructurado

  No se establece ninguna entrada SQLVAR y SQLDOUBLED se establece en blanco. Se emite un aviso SQLCODE +238 (SQLSTATE 01005) (sin tener en cuenta el valor de la opción de vinculación SQLWARN).

  Asigne las 2*SQLD SQLVAR para una operación DESCRIBE satisfactoria.

Las referencias de las listas anteriores a entradas LOB incluyen las entradas de tipo diferenciado cuyo tipo fuente es un tipo LOB.

La opción SQLWARN del mandato BIND o PREP se utiliza para controlar si DESCRIBE (o PREPARE INTO) devolverá los SQLCODE de aviso +236, +237, +239. Se recomienda que el código de la aplicación tenga siempre en cuenta que podrían devolverse estos SQLCODE. El SQLCODE de aviso +238 siempre se devuelve cuando hay entradas LOB o de tipo estructurado en la lista de selección y no hay suficientes SQLVAR en la SQLDA. Es la única manera de que la aplicación pueda saber el número de estructuras SQLVAR que deben doblarse debido a que la existencia de una entrada LOB o de tipo estructurado en el conjunto resultante.

Si se está describiendo una entrada de tipo estructurado, pero no se define ninguna transformación FROM SQL (porque no se especificó ningún TRANSFORM GROUP utilizando el registro especial CURRENT DEFAULT TRANSFORM GROUP (SQLSTATE 42741), o porque el grupo mencionado no tiene una función de transformación FROM SQL definida (SQLSTATE 42744), DESCRIBE emitirá un error. Este error es el mismo que se devuelve para una estructura DESCRIBE de una tabla con una entrada de tipo estructurado.

SQLTYPE y SQLLEN

La Tabla 38 muestra los valores que pueden aparecer en los campos SQLTYPE y SQLLEN de la SQLDA. En DESCRIBE y PREPARE INTO, un valor par de SQLTYPE significa que la columna no permite nulos y un valor impar significa que la columna permite nulos. En FETCH, OPEN y EXECUTE, un valor par de SQLTYPE significa que no se proporciona una variable indicadora y un valor impar significa que SQLIND contiene la dirección de una variable indicadora.

Tabla 38. Valores SQLTYPE y SQLLEN para DESCRIBE, FETCH, OPEN y EXECUTE

	Para DESCRIBE y PREPARE INTO		**Para FETCH, OPEN y EXECUTE**	
SQLTYPE	**Tipo de datos de columna**	**SQLLEN**	**Tipo de datos de variable del lenguaje principal**	**SQLLEN**
384/385	date	10	representación de una fecha en serie de caracteres de longitud fija	atributo de longitud de la variable del lenguaje principal
388/389	time	8	representación de una hora en serie de caracteres de longitud fija	atributo de longitud de la variable del lenguaje principal
392/393	timestamp	26	representación de una indicación de fecha y hora en serie de caracteres de longitud fija	atributo de longitud de la variable del lenguaje principal
396/397	DATALINK	atributo de longitud de la columna	DATALINK	atributo de longitud de la variable del lenguaje principal
400/401	N/D	N/D	serie gráfica terminada en NUL	atributo de longitud de la variable del lenguaje principal
404/405	BLOB	0 *	BLOB	No utilizado. *
408/409	CLOB	0 *	CLOB	No utilizado. *
412/413	DBCLOB	0 *	DBCLOB	No utilizado. *

Tabla 38. Valores SQLTYPE y SQLLEN para DESCRIBE, FETCH, OPEN y EXECUTE (continuación)

	Para DESCRIBE y PREPARE INTO		Para FETCH, OPEN y EXECUTE	
SQLTYPE	**Tipo de datos de columna**	**SQLLEN**	**Tipo de datos de variable del lenguaje principal**	**SQLLEN**
448/449	serie de caracteres de longitud variable	atributo de longitud de la columna	serie de caracteres de longitud variable	atributo de longitud de la variable del lenguaje principal
452/453	serie de caracteres de longitud fija	atributo de longitud de la columna	serie de caracteres de longitud fija	atributo de longitud de la variable del lenguaje principal
456/457	serie larga de caracteres de longitud variable	atributo de longitud de la columna	serie larga de caracteres de longitud variable	atributo de longitud de la variable del lenguaje principal
460/461	N/D	N/D	serie de caracteres terminada en NUL	atributo de longitud de la variable del lenguaje principal
464/465	serie gráfica de longitud variable	atributo de longitud de la columna	serie gráfica de longitud variable	atributo de longitud de la variable del lenguaje principal
468/469	serie gráfica de longitud fija	atributo de longitud de la columna	serie gráfica de longitud fija	atributo de longitud de la variable del lenguaje principal
472/473	serie gráfica de longitud variable larga	atributo de longitud de la columna	serie gráfica larga	atributo de longitud de la variable del lenguaje principal
480/481	coma flotante	8 para precisión doble, 4 para precisión simple	coma flotante	8 para precisión doble, 4 para precisión simple
484/485	decimal empaquetado	precisión en byte 1; escala en byte 2	decimal empaquetado	precisión en byte 1; escala en byte 2
492/493	entero superior	8	entero superior	8
496/497	entero grande	4	entero grande	4
500/501	entero pequeño	2	entero pequeño	2
916/917	No aplicable	No aplicable	variable de referencia a archivos BLOB.	267
920/921	No aplicable	No aplicable	variable de referencia a archivos CLOB.	267
924/925	No aplicable	No aplicable	variable de referencia a archivos DBCLOB.	267
960/961	No aplicable	No aplicable	localizador de BLOB	4
964/965	No aplicable	No aplicable	localizador de CLOB	4
968/969	No aplicable	No aplicable	localizador de DBCLOB	4

Nota:

- El campo len.sqllonglen de la SQLVAR secundaria contiene el atributo de longitud de la columna.
- SQLTYPE se ha modificado desde la versión anterior para portabilidad en DB2. Los valores de las versiones anteriores (consulte Referencia SQL de la versión anterior) seguirán siendo soportados.

SQLTYPE no reconocidos y no soportados

Los valores que aparecen en el campo SQLTYPE de SQLDA dependen del nivel de soporte de tipo de datos disponible en el encargado de enviar los datos así como en el que los recibe. Esto es especialmente importante cuando se añaden nuevos tipos de datos al producto.

Los tipos de datos nuevos pueden recibir o no soporte del que envía los datos o del que los recibe y pueden estar reconocidos o incluso no estarlo por el que envía los datos o el que los recibe. Según la situación, puede devolverse el tipo de datos nuevo o puede devolverse un tipo de datos compatible con el acuerdo del que envía los datos y del que los recibe o bien puede dar como resultado un error.

Cuando el que envía los datos y el que los recibe se ponen de acuerdo para utilizar un tipo de datos compatible, la indicación siguiente expresa la correlación que tendrá lugar. Esta correlación tendrá lugar cuando, como mínimo, o el que envía los datos o el que los recibe no dé soporte al tipo de datos proporcionado. Tanto la aplicación como el gestor de bases de datos pueden proporcionar el tipo de datos no soportado.

Tipo de datos	Tipo de datos compatible
BIGINT	DECIMAL(19, 0)
ROWID[1]	VARCHAR(40) FOR BIT DATA

[1] DB2 Universal Database para z/OS y OS/390 Versión 6 proporciona soporte a ROWID.

Tenga en cuenta que en SQLDA no se proporciona ninguna indicación de que se sustituya el tipo de datos.

Números decimales empaquetados

Los números decimales empaquetados se almacenan en una variación de la notación Decimal codificado en binario (BCD). En BCD, cada nybble (cuatro bits) representa un dígito decimal. Por ejemplo, 0001 0111 1001 representa 179. Por lo tanto, se lee un valor decimal empaquetado nybble a nybble. Se almacena el valor en bytes y después se lee estos bytes en representación hexadecimal para volver a decimal. Por ejemplo, 0001 0111 1001 se convierte en 00000001 01111001 en la representación binaria. Leyendo este número como hexadecimal, se convierte en 0179.

La coma decimal se determina por la escala. En el caso de una columna DEC(12,5), por ejemplo, los 5 dígitos más a la derecha están a la derecha de la coma decimal.

El signo lo indica un nybble a la derecha de los nybbles que representa los dígitos. Un signo positivo o negativo se indica de la siguiente manera:

Tabla 39. Valores para el indicador de signo de un número decimal empaquetado

	Representación		
Signo	**Binaria**	**Decimal**	**Hexadecimal**
Positivo (+)	1100	12	C
Negativo (-)	1101	13	D

En resumen:

- Para almacenar cualquier valor, asigne $p/2+1$ bytes, donde p es la precisión.

- Asigne los nybbles de izquierda a derecha para representar el valor. Si un número tiene una precisión par, se añade un nybble de cero inicial. Esta asignación incluye los dígitos cero iniciales (sin significado) y de cola (significativos).
- El nybble de signo será el segundo nybble del último byte.

Por ejemplo:

Columna	Valor	Nybbles en hexadecimal agrupados por bytes
DEC(8,3)	6574,23	00 65 74 23 0C
DEC(6,2)	-334,02	00 33 40 2D
DEC(7,5)	5,2323	05 23 23 0C
DEC(5,2)	-23,5	02 35 0D

Campo SQLLEN para decimales

El campo SQLLEN contiene la precisión (primer byte) y la escala (segundo byte) de la columna decimal. Al escribir una aplicación portátil, los bytes de la precisión y de la escala se deben establecer individualmente, en lugar de establecerlos conjuntamente como un entero corto. Esto evitará los problemas en la inversión de bytes de enteros.

Por ejemplo, en C:

```
((char *)&(sqlda->sqlvar[i].sqllen))[0] = precision;
((char *)&(sqlda->sqlvar[i].sqllen))[1] = scale;
```

Información relacionada:

- "CHAR" en la página 304

Apéndice D. Vistas de catálogo

Este apéndice contiene una descripción de cada vista de catálogo del sistema, incluyendo los nombres de columna y los tipos de datos.

Vistas de catálogo del sistema

El gestor de bases de datos crea y mantiene dos conjuntos de vistas de catálogo del sistema que se definen además de las tablas base de catálogo del sistema.

- Las vistas SYSCAT son vistas de catálogo de sólo lectura que se encuentran en el esquema SYSCAT. Por omisión, se otorga a PUBLIC el privilegio SELECT sobre estas vistas.
- Las vistas SYSSTAT son vistas de catálogo actualizables que se encuentran en el esquema SYSSTAT. Las vistas actualizables contienen información estadística utilizada por el optimizador. Es posible cambiar los valores de algunas de las columnas de estas vistas para comprobar el rendimiento. (Antes de cambiar algún dato estadístico, se recomienda invocar el mandato RUNSTATS para que toda la información estadística refleje el estado actual.) Las aplicaciones deben escribirse en las vistas SYSSTAT en lugar de en las tablas base de catálogo.

Todas las vistas de catálogo del sistema se crean durante la creación de la base de datos. Las vistas de catálogo no pueden crearse ni eliminarse explícitamente. Las vistas se actualizan durante la actividad normal, en respuesta a sentencias de definición de datos SQL, rutinas de entorno y determinados programas de utilidad. Los datos de las vistas de catálogo del sistema están disponibles mediante las funciones de la consulta SQL normal. Las vistas de catálogo del sistema (a excepción de algunas vistas de catálogo que se puedan actualizar) no se pueden modificar utilizando sentencias de manipulación de datos SQL normales.

Un objeto (tabla, columna, función o índice) sólo aparecerá en la vista de catálogo actualizable de un usuario si dicho usuario ha creado el objeto, posee el privilegio CONTROL sobre el objeto o posee la autoridad DBADM explícita.

El orden de las columnas en las vistas puede cambiar de un release a otro. Para evitar que esto afecte a la lógica de programación, las columnas deben especificarse en una lista de selección explícitamente y debe evitarse la utilización de `SELECT *`. Las columnas tienen nombres coherentes basados en los tipos de objetos que describen.

Objeto descrito	Nombres de columna
Tabla	TABSCHEMA, TABNAME
Índice	INDSCHEMA, INDNAME
Vista	VIEWSCHEMA, VIEWNAME
Restricción	CONSTSCHEMA, CONSTNAME
Activador	TRIGSCHEMA, TRIGNAME
Paquete	PKGSCHEMA, PKGNAME
Tipo	TYPESCHEMA, TYPENAME, TYPEID

Función	ROUTINESCHEMA, ROUTINENAME, ROUTINEID
Método	ROUTINESCHEMA, ROUTINENAME, ROUTINEID
Procedimiento	ROUTINESCHEMA, ROUTINENAME, ROUTINEID
Columna	COLNAME
Esquema	SCHEMANAME
Espacio de tablas	TBSPACE
Grupo de particiones de base de datos	NGNAME
Agrupación de almacenamientos intermedios	BPNAME
Supervisor de sucesos	EVMONNAME
Indicación de la fecha y hora de creación	CREATE_TIME

Guía básica para las vistas de catálogo

Tabla 40. Guía básica para las vistas de catálogo de sólo lectura

Descripción	Vista de catálogo
atributos de tipos de datos estructurados	"SYSCAT.ATTRIBUTES" en la página 556
autorizaciones en base de datos	"SYSCAT.DBAUTH" en la página 578
configuración de agrupación de almacenamiento intermedio en grupo de particiones de base de datos	"SYSCAT.BUFFERPOOLS" en la página 558
tamaño de agrupación de almacenamiento intermedio en partición de base de datos	"SYSCAT.BUFFERPOOLDBPARTITIONS" en la página 557
funciones de conversión	"SYSCAT.CASTFUNCTIONS" en la página 559
restricciones de comprobación	"SYSCAT.CHECKS" en la página 560
privilegios de columna	"SYSCAT.COLAUTH" en la página 561
columnas	"SYSCAT.COLUMNS" en la página 569
columnas referenciadas por restricciones de comprobación	"SYSCAT.COLCHECKS" en la página 562
columnas utilizadas en dimensiones	"SYSCAT.COLUSE" en la página 574
columnas utilizadas en claves	"SYSCAT.KEYCOLUSE" en la página 605
dependencias de restricción	"SYSCAT.CONSTDEP" en la página 575
particiones de base de datos de grupo de particiones de base de datos	"SYSCAT.DBPARTITIONGROUPDEF" en la página 580
definiciones de grupo de particiones de base de datos	"SYSCAT.DBPARTITIONGROUPS" en la página 581
tipos de datos	"SYSCAT.DATATYPES" en la página 576
estadísticas detalladas de grupo de columnas	"SYSCAT.COLGROUPDIST" en la página 564
	"SYSCAT.COLGROUPDISTCOUNTS" en la página 565
	"SYSCAT.COLGROUPS" en la página 566
opciones de columna detalladas	"SYSCAT.COLOPTIONS" en la página 568
estadísticas detalladas de columna	"SYSCAT.COLDIST" en la página 563
definiciones de supervisor de sucesos	"SYSCAT.EVENTMONITORS" en la página 582
sucesos supervisados actualmente	"SYSCAT.EVENTS" en la página 584
	"SYSCAT.EVENTS" en la página 585
dependencias de función[1]	"SYSCAT.ROUTINEDEP" en la página 623
correlación de funciones	"SYSCAT.FUNCMAPPINGS" en la página 589
opciones de correlación de funciones	"SYSCAT.FUNCMAPOPTIONS" en la página 587
opciones de correlación de parámetros de función	"SYSCAT.FUNCMAPPARMOPTIONS" en la página 588
parámetros de función[1]	"SYSCAT.ROUTINEPARMS" en la página 624
funciones[1]	"SYSCAT.ROUTINES" en la página 626

Tabla 40. Guía básica para las vistas de catálogo de sólo lectura (continuación)

Descripción	Vista de catálogo
jerarquías (tipos, tablas, vistas)	"SYSCAT.HIERARCHIES" en la página 590
	"SYSCAT.FULLHIERARCHIES" en la página 586
columnas de identidad	"SYSCAT.COLIDENTATTRIBUTES" en la página 567
columnas de índice	"SYSCAT.INDEXCOLUSE" en la página 592
dependencias de índice	"SYSCAT.INDEXDEP" en la página 593
explotación del índice	"SYSCAT.INDEXEXPLOITRULES" en la página 599
dependencias de extensión de índice	"SYSCAT.INDEXEXTENSIONDEP" en la página 600
parámetros de extensión de índice	"SYSCAT.INDEXEXTENSIONPARMS" en la página 602
métodos de búsqueda de extensiones de índice	"SYSCAT.INDEXEXTENSIONMETHODS" en la página 601
extensiones de índice	"SYSCAT.INDEXEXTENSIONS" en la página 603
opciones de índice	"SYSCAT.INDEXOPTIONS" en la página 604
privilegios de índice	"SYSCAT.INDEXAUTH" en la página 591
índices	"SYSCAT.INDEXES" en la página 594
dependencias de método[1]	"SYSCAT.ROUTINEDEP" en la página 623
parámetros de método[1]	"SYSCAT.ROUTINES" en la página 626
métodos[1]	"SYSCAT.ROUTINES" en la página 626
correlación de objeto	"SYSCAT.NAMEMAPPINGS" en la página 606
dependencias de paquete	"SYSCAT.PACKAGEDEP" en la página 608
privilegios de paquete	"SYSCAT.PACKAGEAUTH" en la página 607
paquetes	"SYSCAT.PACKAGES" en la página 609
correlaciones de particiones	"SYSCAT.PARTITIONMAPS" en la página 614
privilegios de paso a través	"SYSCAT.PASSTHRUAUTH" en la página 615
especificaciones de predicado	"SYSCAT.PREDICATESPECS" en la página 616
opciones de procedimiento	"SYSCAT.PROCOPTIONS" en la página 617
opciones de parámetro de procedimiento	"SYSCAT.PROCPARMOPTIONS" en la página 618
parámetros de procedimiento[1]	"SYSCAT.ROUTINEPARMS" en la página 624
procedimientos[1]	"SYSCAT.ROUTINES" en la página 626
proporciona DB2 Universal Database para compatibilidad con z/OS y OS/390	"SYSIBM.SYSDUMMY1" en la página 555
restricciones de referencia	"SYSCAT.REFERENCES" en la página 619
opciones de tabla remota	"SYSCAT.TABOPTIONS" en la página 649

Tabla 40. Guía básica para las vistas de catálogo de sólo lectura (continuación)

Descripción	Vista de catálogo
correlación de tipos de datos invertida	"SYSCAT.REVTYPEMAPPINGS" en la página 620
dependencias de rutina	"SYSCAT.ROUTINEDEP" en la página 623
parámetros de rutina[1]	"SYSCAT.ROUTINEPARMS" en la página 624
privilegios de rutina	"SYSCAT.ROUTINEAUTH" en la página 622
rutinas[1]	"SYSCAT.ROUTINES" en la página 626
privilegios de esquema	"SYSCAT.SCHEMAAUTH" en la página 632
esquemas	"SYSCAT.SCHEMATA" en la página 633
privilegios de secuencia	"SYSCAT.SEQUENCEAUTH" en la página 634
secuencias	"SYSCAT.SEQUENCES" en la página 635
opciones de servidor	"SYSCAT.SERVEROPTIONS" en la página 636
valores de opciones de servidor	"SYSCAT.USEROPTIONS" en la página 656
sentencias en paquetes	"SYSCAT.STATEMENTS" en la página 638
procedimientos almacenados	"SYSCAT.ROUTINES" en la página 626
servidores del sistema	"SYSCAT.SERVERS" en la página 637
restricciones de tabla	"SYSCAT.TABCONST" en la página 641
dependencias de tabla	"SYSCAT.TABDEP" en la página 642
privilegios de tabla	"SYSCAT.TABAUTH" en la página 639
privilegios de uso de espacios de tablas	"SYSCAT.TBSPACEAUTH" en la página 650
espacios de tablas	"SYSCAT.TABLESPACES" en la página 648
tablas	"SYSCAT.TABLES" en la página 643
transformaciones	"SYSCAT.TRANSFORMS" en la página 651
dependencias de activador	"SYSCAT.TRIGDEP" en la página 652
activadores	"SYSCAT.TRIGGERS" en la página 653
correlación de tipos	"SYSCAT.TYPEMAPPINGS" en la página 654
funciones definidas por el usuario	"SYSCAT.ROUTINES" en la página 626
vistas	"SYSCAT.TABLES" en la página 643
	"SYSCAT.VIEWS" en la página 657
opciones de reiniciador	"SYSCAT.WRAPOPTIONS" en la página 658
reiniciadores	"SYSCAT.WRAPPERS" en la página 659

[1] Todavía existen las vistas de catálogo para funciones, métodos y procedimientos de DB2 Versión 7.1 y anterior. Sin embargo, estas vistas no reflejan ningún cambio desde DB2 Versión 7.1. Las vistas son las siguientes:

```
Funciones:      SYSCAT.FUNCTIONS, SYSCAT.FUNCDEP, SYSCAT.FUNCPARMS
Métodos:        SYSCAT.FUNCTIONS, SYSCAT.FUNCDEP, SYSCAT.FUNCPARMS
Procedimientos: SYSCAT.PROCEDURES, SYSCAT.PROCPARMS
```

Guía básica para las vistas de catálogo

Tabla 41. Guía básica para las vistas de catálogo actualizables

Descripción	Vista de catálogo
columnas	"SYSSTAT.COLUMNS" en la página 662
estadísticas detalladas de columna	"SYSSTAT.COLDIST" en la página 660
índices	"SYSSTAT.INDEXES" en la página 664
rutinas[1]	"SYSSTAT.ROUTINES" en la página 668
tablas	"SYSSTAT.TABLES" en la página 670

[1] La vista de catálogo SYSSTAT.FUNCTIONS todavía existe para actualizar los datos estadísticos de funciones y métodos. Sin embargo, esta vista no refleja ningún cambio desde DB2 Versión 7.1.

SYSIBM.SYSDUMMY1

Contiene una fila. Esta vista está disponible para las aplicaciones que necesitan compatibilidad con DB2 Universal Database para z/OS y OS/390.

Tabla 42. Vista de catálogo SYSCAT.DUMMY1

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
IBMREQD	CHAR(1)		Y

SYSCAT.ATTRIBUTES

Contiene una fila para cada atributo (incluidos los atributos heredados donde sea aplicable) que se define para un tipo de datos estructurado definido por el usuario.

Tabla 43. Vista de catálogo SYSCAT.ATTRIBUTES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TYPESCHEMA	VARCHAR(128)		Nombre calificado del tipo de datos estructurado que incluye el atributo.
TYPENAME	VARCHAR(128)		
ATTR_NAME	VARCHAR(128)		Nombre del atributo.
ATTR_TYPESCHEMA	VARCHAR(128)		Nombre calificado del tipo del atributo.
ATTR_TYPENAME	VARCHAR(128)		
TARGET_TYPESCHEMA	VARCHAR(128)	Sí	Nombre calificado del tipo de destino si el tipo del atributo es REFERENCE. Valor nulo si el tipo del atributo no es REFERENCE.
TARGET_TYPENAME	VARCHAR(128)	Sí	
SOURCE_TYPESCHEMA	VARCHAR(128)		Nombre calificado del tipo de datos en la jerarquía de tipos de datos en que se ha introducido el atributo. Para atributos no heredado, estas columnas son igual que TYPESCHEMA y TYPENAME.
SOURCE_TYPENAME	VARCHAR(128)		
ORDINAL	SMALLINT		Posición del atributo en la definición del tipo de datos estructurado, empezando por cero.
LENGTH	INTEGER		Longitud máxima de datos; 0 para tipos diferenciados. La columna LENGTH indica la precisión para los campos DECIMAL.
SCALE	SMALLINT		Escala para los campos DECIMAL; 0 si no es DECIMAL.
CODEPAGE	SMALLINT		Página de códigos del atributo. Para los atributos de series de caracteres no definidos con FOR BIT DATA, el valor es la página de códigos de la base de datos. Para los atributos de series gráficas, el valor es la página de códigos DBCS implícita en la página de códigos de la base de datos (compuesta). De lo contrario, el valor es 0.
LOGGED	CHAR(1)		Sólo se aplica a los atributos cuyo tipo sea LOB o diferenciado basado en LOB; de lo contrario, está en blanco. Y = El atributo se anota cronológicamente. N = El atributo no se anota cronológicamente.
COMPACT	CHAR(1)		Sólo se aplica a los atributos cuyo tipo sea LOB o diferenciado basado en LOB; de lo contrario, está en blanco. Y = El atributo se compacta en el almacenamiento. N = El atributo no se compacta.
DL_FEATURES	CHAR(10)		Sólo se aplica a los atributos de tipo DATALINK. Está en blanco para los atributos de tipo REFERENCE; de lo contrario, es un valor nulo. Codifica diversas características de DATALINK como, por ejemplo, tipo de enlace, modalidad de control, recuperación y propiedades de desenlace.

SYSCAT.BUFFERPOOLDBPARTITIONS

Contiene una fila para cada partición de base de datos en el agrupamiento de almacenamientos intermedios para la que el tamaño de la agrupación de almacenamientos intermedios de la partición de base de datos sea diferente del tamaño por omisión de la columna NPAGES de SYSCAT.BUFFERPOOLS.

Tabla 44. Vista de catálogo SYSCAT.BUFFERPOOLDBPARTITIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
BUFFERPOOLID	INTEGER		Identificador interno de agrupación de almacenamientos intermedios
DBPARTITIONNUM	SMALLINT		El número de partición de base de datos
NPAGES	INTEGER		El número de páginas de esta agrupación de almacenamientos intermedios en esta partición de base de datos

SYSCAT.BUFFERPOOLS

Contiene una fila para cada agrupación de almacenamientos intermedios de cada grupo de particiones de base de datos.

Tabla 45. Vista de catálogo SYSCAT.BUFFERPOOLS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
BPNAME	VARCHAR(128)		Nombre de la agrupación de almacenamientos intermedios
BUFFERPOOLID	INTEGER		Identificador interno de agrupación de almacenamientos intermedios
NGNAME	VARCHAR(128)	Sí	Nombre del grupo de particiones de base de datos (NULL si la agrupación de almacenamientos intermedios existe en todas las particiones de base de datos de la base de datos)
NPAGES	INTEGER		El número de páginas de la agrupación de almacenamientos intermedios
PAGESIZE	INTEGER		Tamaño de página para esta agrupación de almacenamientos intermedios
ESTORE	CHAR(1)		N = Esta agrupación de almacenamientos intermedios no utiliza el almacenamiento ampliado. Y = Esta agrupación de almacenamientos intermedios utiliza almacenamiento ampliado.

SYSCAT.CASTFUNCTIONS

Contiene una fila para cada función de conversión. No incluye funciones de conversión incorporadas.

Tabla 46. Vista de catálogo SYSCAT.CASTFUNCTIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
FROM_TYPESCHEMA	VARCHAR(128)		Nombre calificado del tipo de datos del parámetro.
FROM_TYPENAME	VARCHAR(128)		
TO_TYPESCHEMA	VARCHAR(128)		Nombre calificado del tipo de datos del resultado después de la conversión.
TO_TYPENAME	VARCHAR(128)		
FUNCSCHEMA	VARCHAR(128)		Nombre calificado de la función.
FUNCNAME	VARCHAR(128)		
SPECIFICNAME	VARCHAR(128)		El nombre de la instancia de función.
ASSIGN_FUNCTION	CHAR(1)		Y = Función de asignación implícita N = No es una función de asignación

SYSCAT.CHECKS

Contiene una fila para cada restricción CHECK.

Tabla 47. Vista de catálogo SYSCAT.CHECKS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
CONSTNAME	VARCHAR(18)		Nombre de la restricción de comprobación (exclusivo dentro de una tabla).
DEFINER	VARCHAR(128)		ID de autorización bajo el que se ha definido la restricción de comprobación.
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla a la que se aplica esta restricción.
TABNAME	VARCHAR(128)		
CREATE_TIME	TIMESTAMP		La hora en la que se ha definido la restricción. Se utiliza para resolver funciones que se utilizan en esta restricción. No se elegirá ninguna función creada tras la definición de la restricción.
QUALIFIER	VARCHAR(128)		Valor del esquema por omisión en el momento de la definición del objeto. Se utiliza para completar cualquier referencia no calificada.
TYPE	CHAR(1)		Tipo de restricción de comprobación: A = Restricción de comprobación generada por el sistema para la columna GENERATED ALWAYS C = Restricción de comprobación F = Dependencia funcional O = La restricción es una propiedad del objeto
FUNC_PATH	VARCHAR(254)		La vía de acceso de SQL actual que se ha utilizado al crear la restricción.
TEXT	CLOB(64K)		El texto de la cláusula CHECK. Consulte la Nota 1.

Notas:

1. En la vista de catálogo, el texto de la cláusula CHECK siempre se muestra en la página de códigos de la base de datos y puede contener caracteres de sustitución. La restricción CHECK siempre se aplicará en la página de códigos de la tabla destino y no contendrá ningún carácter de sustitución cuando se aplique. (La restricción CHECK se aplicará en base al texto original de la página de códigos de la tabla destino, que es posible que no incluya los caracteres de sustitución.)

SYSCAT.COLAUTH

Contiene una o varias filas para cada usuario o grupo a los que se ha otorgado un privilegio a nivel de columna, que indican el tipo de privilegio y si se puede otorgar o no.

Tabla 48. Vista de catálogo SYSCAT.COLAUTH

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
GRANTOR	VARCHAR(128)		ID de autorización del usuario que ha otorgado los privilegios o SYSIBM.
GRANTEE	VARCHAR(128)		ID de autorización del usuario o grupo que tiene los privilegios.
GRANTEETYPE	CHAR(1)		U = Se otorga a un usuario individual. G = Se otorga a un grupo.
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla o vista.
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		Nombre de la columna a la que se aplica este privilegio.
COLNO	SMALLINT		El número de esta columna en la tabla o vista.
PRIVTYPE	CHAR(1)		Indica el tipo de privilegio que se tiene en la tabla o vista: U = Actualiza privilegio R = Hace referencia a privilegio
GRANTABLE	CHAR(1)		Indica si se puede otorgar el privilegio. G = Otorgable N = No otorgable

SYSCAT.COLCHECKS

Cada fila representa alguna columna a la que una restricción CHECK hace referencia.

Tabla 49. Vista de catálogo SYSCAT.COLCHECKS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
CONSTNAME	VARCHAR(18)		Nombre de la restricción de comprobación. (Exclusivo en una tabla. Puede ser generado por el sistema.)
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla que contiene la columna a la que se hace referencia.
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		Nombre de columna.
USAGE	CHAR(1)		D = Columna hija con dependencia funcional P = Columna padre con dependencia funcional R = Se hace referencia a la columna en la restricción de comprobación. S = La columna es una columna fuente en la restricción de comprobación generada por el sistema que da soporte a una columna generada. T = La columna es una columna destino en la restricción de comprobación generada por el sistema que da soporte a una columna generada.

SYSCAT.COLDIST

Contiene estadísticas detalladas de columna para que las utilice el optimizador. Cada fila describe el valor N más frecuente de algunas columnas.

Tabla 50. Vista de catálogo SYSCAT.COLDIST

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla a la que se aplica esta entrada.
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		Nombre de la columna a la que se aplica esta entrada.
TYPE	CHAR(1)		F = Frecuencia (valor más frecuente) Q = Valor cuantil
SEQNO	SMALLINT		• Si TYPE = F, entonces N en esta columna identifica el valor N más frecuente. • Si TYPE=Q, entonces N en esta columna identifica el valor N cuantil.
COLVALUE	VARCHAR(254)	Sí	El valor de datos, como un literal de caracteres o un valor nulo. Consulte la Nota 1.
VALCOUNT	BIGINT		• Si TYPE = F, entonces VALCOUNT es el número de ocurrencias de COLVALUE en la columna. • Si TYPE = Q, entonces VALCOUNT es el número de filas cuyo valor es menor o igual que COLVALUE.
DISTCOUNT	BIGINT	Sí	Si TYPE = Q, esta columna registra el número de valores diferenciados que son menores o iguales que COLVALUE (nulo si no está disponible).

Notas:

1. En la vista de catálogo, el valor de COLVALUE siempre se muestra en la página de códigos de la base de datos y puede contener caracteres de sustitución. Sin embargo, las estadísticas se reúnen internamente en la página de códigos de la tabla de la columna y, por tanto, utilizarán los valores reales de la columna cuando se apliquen durante la optimización de la consulta.

SYSCAT.COLGROUPDIST

Contiene una fila para cada valor de una columna de un grupo de columnas que constituye el valor *n* más frecuente del grupo de columnas o el valor cuantil *n* del grupo de columnas.

Tabla 51. Vista de catálogo SYSCAT.COLGROUPDIST

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
COLGROUPID	INTEGER		Identificador interno del grupo de columnas.
TYPE	CHAR(1)		F = Valor de frecuencia Q = Valor cuantil
ORDINAL	SMALLINT		El número de orden de la columna dentro del grupo.
SEQNO	SMALLINT		El número de secuencia *n* que representa el valor *n* de TYPE.
COLVALUE	VARCHAR(254)	Sí	El valor de los datos como un literal de caracteres o un valor nulo.

SYSCAT.COLGROUPDISTCOUNTS

Contiene una fila para los datos estadísticos de distribución que se aplican al valor *n* más frecuente de un grupo de columnas o el valor cuantil *n* del grupo de columnas.

Tabla 52. Vista de catálogo SYSCAT.COLGROUPDISTCOUNTS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
COLGROUPID	INTEGER		Identificador interno del grupo de columnas.
TYPE	CHAR(1)		F = Valor de frecuencia Q = Valor cuantil
SEQNO	SMALLINT		El número de secuencia *n* que representa el valor *n* de TYPE.
VALCOUNT	BIGINT		Si TYPE = F, VALCOUNT es el número de ocurrencias de COLVALUE para el grupo de columnas con este SEQNO. Si TYPE = Q, VALCOUNT es el número de filas cuyo valor es menor o igual que COLVALUE para el grupo de columnas con este SEQNO.
DISTCOUNT	BIGINT	Sí	Si TYPE = Q, esta columna registra el número de valores diferenciados que son menores o iguales que COLVALUE para el grupo de columnas con este SEQNO (nulo si no está disponible).

SYSCAT.COLGROUPS

Contiene una fila para cada grupo de columnas y los datos estadísticos que se aplican a todo el grupo de columnas.

Tabla 53. Vista de catálogo SYSCAT.COLGROUPS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
COLGROUPSCHEMA	VARCHAR(128)		Nombre calificado del grupo de columnas.
COLGROUPNAME	VARCHAR(128)		
COLGROUPID	INTEGER		Identificador interno del grupo de columnas.
COLGROUPCARD	BIGINT		Cardinalidad del grupo de columnas.
NUMFREQ_VALUES	SMALLINT		Número de valores frecuentes recopilados para el grupo de columnas.
NUMQUANTILES	SMALLINT		Número de valores cuantiles recopilados para el grupo de columnas.

SYSCAT.COLIDENTATTRIBUTES

Contiene una fila para cada columna de identidad que se define para una tabla.

Tabla 54. Columnas de la vista de catálogo SYSCAT.COLIDENTATTRIBUTES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla o vista que contiene la columna.
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		Nombre de columna.
START	DECIMAL(31,0)		Valor inicial.
INCREMENT	DECIMAL(31,0)		Valor de incremento.
MINVALUE	DECIMAL(31,0)		Valor mínimo.
MAXVALUE	DECIMAL(31,0)		Valor máximo.
CYCLE	CHAR(1)		Indica si se producirá el ciclo cuando se alcance un límite: Y - se producirá el ciclo N - no se producirá el ciclo
CACHE	INTEGER		El número de valores de secuencia que se deben preasignar en memoria para el acceso más rápido. 0 indica que no se preasignan valores.
SEQID	INTEGER		ID interno de la secuencia.

SYSCAT.COLOPTIONS

Cada fila contiene los valores de las opciones específicas de columna.

Tabla 55. Vista de catálogo SYSCAT.COLOPTIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TABSCHEMA	VARCHAR(128)		Apodo calificado para la columna.
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		Nombre de columna local.
OPTION	VARCHAR(128)		Nombre de la opción de columna.
SETTING	VARCHAR(255)		Valor de la opción de columna.

SYSCAT.COLUMNS

Contiene una fila para cada columna (incluidas las columnas heredadas, si es aplicable) que se define para una tabla o vista. Todas las vistas de catálogo tienen entradas en la tabla SYSCAT.COLUMNS.

Tabla 56. Vista de catálogo SYSCAT.COLUMNS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla o vista que contiene la columna.
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		Nombre de columna.
COLNO	SMALLINT		Lugar numérico de la columna en la tabla o vista, empezando por cero.
TYPESCHEMA	VARCHAR(128)		Contiene el nombre calificado del tipo, si el tipo de datos de la columna es diferenciado. De lo contrario, TYPESCHEMA contiene el valor SYSIBM y TYPENAME contiene el tipo de datos de la columna (en formato largo, por ejemplo, CHARACTER). Si se especifica FLOAT o FLOAT (*n*) siendo *n* mayor que 24, TYPENAME se redenomina a DOUBLE. Si se especifica FLOAT(*n*) siendo *n* menor que 25, TYPENAME se redenomina a REAL. También, NUMERIC se redenomina por DECIMAL.
TYPENAME	VARCHAR(18)		
LENGTH	INTEGER		Longitud máxima de datos. 0 para tipos diferenciados. La columna LENGTH indica la precisión para los campos DECIMAL.
SCALE	SMALLINT		Escala para los campos DECIMAL; 0 si no es DECIMAL.
DEFAULT	VARCHAR(254)	Sí	El valor por omisión de la columna de una tabla expresado como una constante, un registro especial o una función de conversión adecuada para el tipo de datos de la columna. También puede ser la palabra clave NULL. Los valores se pueden convertir de lo que se ha especificado como valor por omisión. Por ejemplo, las constantes de fecha y hora se presentan en formato ISO y los nombres de función de conversión se califican con el nombre de esquema y los identificadores son delimitados (vea la Nota 3). El valor nulo si no se ha especificado una cláusula DEFAULT o la columna es una columna de vista.

Tabla 56. Vista de catálogo SYSCAT.COLUMNS (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
NULLS	CHAR(1)		Y = La columna puede contener nulos N = La columna no puede contener nulos. El valor puede ser N para una columna de vista que se deriva de una expresión o función. No obstante, estas columnas permiten nulos cuando la sentencia que utiliza la vista se procesa con avisos para errores aritméticos. Consulte la Nota 1.
CODEPAGE	SMALLINT		Página de códigos de la columna. Para las columnas de series de caracteres no definidas con el atributo FOR BIT DATA, el valor es la página de códigos de base de datos. Para columnas de series gráficas, el valor es la página de códigos DBCS implícita en la página de códigos de la base de datos (compuesta). De lo contrario, el valor es 0.
LOGGED	CHAR(1)		Sólo se aplica a las columnas cuyo tipo sea LOB o un tipo diferenciado basado en LOB (está en blanco de lo contrario). Y = La columna se anota cronológicamente. N = La columna no se anota cronológicamente.
COMPACT	CHAR(1)		Sólo se aplica a las columnas cuyo tipo sea LOB o un tipo diferenciado basado en LOB (está en blanco de lo contrario). Y = La columna se compacta en el almacenamiento. N = La columna no se compacta.
COLCARD	BIGINT		El número de valores diferenciados en la columna; –1 si no se reúnen estadísticas; –2 para columnas heredadas y columnas de tablas-H.
HIGH2KEY	VARCHAR(254)	Sí	Segundo nivel más alto de la columna. Este campo está vacío si no se reúnen estadísticas y para columnas heredadas y columnas de tablas-H. Consulte las notas 2 y 4.
LOW2KEY	VARCHAR(254)	Sí	Segundo nivel más bajo de la columna. Este campo está vacío si no se reúnen estadísticas y para columnas heredadas y columnas de tablas-H. Consulte las notas 2 y 4.
AVGCOLLEN	INTEGER		Promedio de espacio necesario para la longitud de columna. –1 si es un campo largo o LOB o bien si no se han reunido estadísticas; –2 para columnas heredadas y columnas de tablas-H.
KEYSEQ	SMALLINT	Sí	La posición numérica de la columna en la clave primaria de la tabla. Este campo es nulo para subtablas y tablas de jerarquía.

Tabla 56. Vista de catálogo SYSCAT.COLUMNS (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
PARTKEYSEQ	SMALLINT	Sí	La posición numérica de la columna en la clave de particionamiento de la tabla. Este campo es nulo ó 0 si la columna no forma parte de la clave de particionamiento. Este campo es también nulo para subtablas y tablas de jerarquía.
NQUANTILES	SMALLINT		El número de valores cuantiles registrados en SYSCAT.COLDIST para esta columna; –1 si no se reúnen estadísticas; –2 para columnas heredadas y columnas de tablas H.
NMOSTFREQ	SMALLINT		El número de los valores más frecuentes registrados en SYSCAT.COLDIST para esta columna; –1 si no se reúnen estadísticas; –2 para columnas heredadas y columnas de tablas H.
NUMNULLS	BIGINT		Contiene el número de nulos en una columna. –1 si no se reúnen estadísticas.
TARGET_TYPESCHEMA	VARCHAR(128)	Sí	Nombre calificado del tipo de destino si el tipo de la columna es REFERENCE. Valor nulo si el tipo de la columna no es REFERENCE.
TARGET_TYPENAME	VARCHAR(18)	Sí	
SCOPE_TABSCHEMA	VARCHAR(128)	Sí	Nombre calificado del ámbito (tabla de destino) si el tipo de la columna es REFERENCE. Valor nulo si el tipo de la columna no es REFERENCE o el ámbito no está definido.
SCOPE_TABNAME	VARCHAR(128)	Sí	
SOURCE_TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla o vista de la jerarquía respectiva en la que se ha introducido la columna. Para columnas no heredadas, los valores son los mismos que TBCREATOR y TBNAME. Es nulo para columnas de vistas y tablas no de tipo.
SOURCE_TABNAME	VARCHAR(128)		
DL_FEATURES	CHAR(10)	Sí	Sólo se aplica a las columnas de tipo DATALINK. De lo contrario, es nulo. Cada posición de carácter se define de la manera siguiente: 1. Tipo de enlace (U para URL) 2. Control de enlace (A para archivo, N para no) 3. Integridad (T para todos, N para ninguno) 4. Permiso de lectura (S para sistema de archivos, B para base de datos) 5. Permiso de escritura (S para sistema de archivos, B para bloqueado, A para administración que necesita un símbolo para su actualización, N para administración que no necesita ningún símbolo para su actualización) 6. Recuperación (S para sí, N para no) 7. En desenlace (R para restaurar, S para suprimir, N para no aplicable) Los caracteres del 8 al 10 se reservan para su utilización en el futuro.

Tabla 56. Vista de catálogo SYSCAT.COLUMNS (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
SPECIAL_PROPS	CHAR(8)	Sí	Sólo se aplica a las columnas de tipo REFERENCE. De lo contrario, es nulo. Cada posición de carácter se define de la manera siguiente: Columna identificador de objeto (OID) (S para sí, N para no) Generada por el usuario o generada por el sistema (U para usuario, S para sistema)
HIDDEN	CHAR(1)		Tipo de columna oculta S = Columna oculta gestionada por el sistema En blanco si la columna no es oculta
INLINE_LENGTH	INTEGER		Longitud de la columna de tipo estructurado que se puede mantener con una fila de la tabla base. 0 si no se define explícitamente mediante una sentencia ALTER/CREATE TABLE.
IDENTITY	CHAR(1)		'S' indica que la columna es una columna de identidad; 'N' indica que la columna no es una columna de identidad.
GENERATED	CHAR(1)		Tipo de columna generada A = El valor de la columna es siempre generada D = El valor de la columna se genera por omisión En blanco si la columna no es generada
COMPRESS	CHAR(1)		S = Valores por omisión del sistema de compresión O = Compresión desactivada
TEXT	CLOB(64K)		Contiene el texto de la columna generada, comenzando con la palabra clave AS.
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario.
AVGDISTINCTPERPAGE	DOUBLE	Sí	Para su utilización en el futuro.
PAGEVARIANCERATIO	DOUBLE	Sí	Para su utilización en el futuro.
SUB_COUNT	SMALLINT		Número medio de subelementos. Aplicable a columnas de caracteres solamente. Por ejemplo, considere la serie siguiente: 'datos simulación analítica inteligencia empresarial'. En este ejemplo, SUB_COUNT = 5, porque hay 5 subelementos en la serie.
SUB_DELIM_LENGTH	SMALLINT		Longitud media de cada delimitador que separa cada subelemento. Aplicable a columnas de caracteres solamente. Por ejemplo, considere la serie siguiente: 'datos simulación analítica inteligencia empresarial'. En este ejemplo, SUB_DELIM_LENGTH = 1, porque cada delimitador es un solo blanco.

Tabla 56. Vista de catálogo SYSCAT.COLUMNS (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción

Notas:

1. Empezando en la Versión 2, el valor D (que indica ningún nulo con valor por omisión) ya no se utiliza. En su lugar, la utilización de WITH DEFAULT se indica por un valor no nulo en la columna DEFAULT.
2. Empezando en la Versión 2, la representación de datos numéricos se ha cambiado a literales de caracteres. El tamaño se ha ampliado de 16 a 33 bytes.
3. Para la Versión 2.1.0, los nombres de función de conversión no se han delimitado y pueden seguir apareciendo de esta manera en la columna DEFAULT. También, algunas columnas de vistas incluyen valores por omisión que todavía aparecen en la columna DEFAULT.
4. En la vista de catálogo, los valores de HIGH2KEY y de LOW2KEY siempre se muestran en la página de códigos de la base de datos y pueden contener caracteres de sustitución. Sin embargo, las estadísticas se reúnen internamente en la página de códigos de la tabla de la columna y, por tanto, utilizarán los valores reales de la columna cuando se apliquen durante la optimización de la consulta.

SYSCAT.COLUSE

Contiene una fila para cada columna que participa en la cláusula DIMENSIONS de la sentencia CREATE TABLE.

Tabla 57. Vista de catálogo SYSCAT.COLUSE

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla que contiene la columna
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		El nombre de la columna
DIMENSION	SMALLINT		El número de dimensiones, basado en el orden de las dimensiones especificadas en la cláusula DIMENSIONS (posición inicial = 0). Para una dimensión compuesta, este valor será el mismo para cada componente de la dimensión.
COLSEQ	SMALLINT		Posición numérica de la columna en la dimensión a la que pertenece (posición inicial = 0). El valor es 0 para una sola columna en una dimensión no compuesta.
TYPE	CHAR(1)		Tipo de dimensión. C = clústeres/clústeres de varias dimensiones (MDC)

SYSCAT.CONSTDEP

Contiene una fila para cada dependencia que tiene una restricción de otro objeto.

Tabla 58. Vista de catálogo SYSCAT.CONSTDEP

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
CONSTNAME	VARCHAR(18)		Nombre de la restricción.
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla a la que se aplica la restricción.
TABNAME	VARCHAR(128)		
BTYPE	CHAR(1)		Tipo de objeto del que depende la restricción. Los valores posibles son: F = Instancia de función I = Instancia de índice R = Tipo estructurado
BSCHEMA	VARCHAR(128)		Nombre calificado del objeto del que depende la restricción.
BNAME	VARCHAR(18)		

SYSCAT.DATATYPES

Contiene una fila para cada tipo de datos, los tipos de datos internos y definidos por el usuario inclusive.

Tabla 59. Vista de catálogo SYSCAT.DATATYPES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TYPESCHEMA	VARCHAR(128)		Nombre calificado del tipo de datos (para tipos internos, TYPESCHEMA es SYSIBM).
TYPENAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		ID de autorización bajo el cual se ha creado el tipo.
SOURCESCHEMA	VARCHAR(128)	Sí	Nombre calificado del tipo fuente para tipos diferenciados. Nombre calificado del tipo interno que se utiliza como el tipo de referencia que sirve de representación para las referencias a tipos estructurados. Nulo para los demás tipos.
SOURCENAME	VARCHAR(18)	Sí	
METATYPE	CHAR(1)		S = Tipo predefinido por el sistema T = Tipo diferenciado R = Tipo estructurado
TYPEID	SMALLINT		Identificador interno del tipo de datos generado por el sistema.
SOURCETYPEID	SMALLINT	Sí	ID de tipo interno del tipo fuente (nulo para tipos internos). Para los tipos estructurados definidos por el usuario, éste es el ID de tipo interno del tipo de representación de referencia.
LENGTH	INTEGER		Longitud máxima del tipo. 0 para tipos con parámetros predefinidos por el sistema (por ejemplo, DECIMAL y VARCHAR). Para los tipos estructurados definidos por el usuario, indica la longitud del tipo de representación de referencia.
SCALE	SMALLINT		Escala para los tipos diferenciados o tipos de representación de referencia basados en el tipo DECIMAL predefinido por el sistema. 0 para los demás tipos (el propio DECIMAL inclusive). Para los tipos estructurados definidos por el usuario, indica la longitud del tipo de representación de referencia.
CODEPAGE	SMALLINT		Página de códigos para tipos diferenciados de caracteres y gráficos o tipos de representación de referencia; de lo contrario, 0.
CREATE_TIME	TIMESTAMP		Tiempo de creación del tipo de datos.
ATTRCOUNT	SMALLINT		El número de atributos en tipo de datos.
INSTANTIABLE	CHAR(1)		Y = El tipo se puede instaciar. N = El tipo no se puede instanciar.

Tabla 59. Vista de catálogo SYSCAT.DATATYPES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
WITH_FUNC_ACCESS	CHAR(1)		Y = Se pueden invocar todos los métodos para este tipo utilizando la función notación. N = No se pueden invocar los métodos para este tipo utilizando la función notación.
FINAL	CHAR(1)		Y = El tipo definido por el usuario no puede tener subtipos. N = El tipo definido por el usuario puede tener subtipos.
INLINE_LENGTH	INTEGER		Longitud de tipo estructurado que se puede mantener con una fila de la tabla base. 0 si no se define explícitamente un valor mediante una sentencia CREATE TYPE.
NATURAL_INLINE_LENGTH	INTEGER		Longitud en línea calculada por el sistema del tipo estructurado.
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario o nulo.

SYSCAT.DBAUTH

Registra las autorizaciones de base de datos que ostentan los usuarios.

Tabla 60. Vista de catálogo SYSCAT.DBAUTH

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
GRANTOR	VARCHAR(128)		SYSIBM o ID de autorización del usuario que ha otorgado los privilegios.
GRANTEE	VARCHAR(128)		ID de autorización del usuario o grupo que tiene los privilegios.
GRANTEETYPE	CHAR(1)		U = Se otorga a un usuario individual. G = Se otorga a un grupo.
DBADMAUTH	CHAR(1)		Si el destinatario de la operación de otorgar tiene la autorización DBADM sobre la base de datos: Y = Mantiene la autoridad. N = No mantiene la autoridad.
CREATETABAUTH	CHAR(1)		Si el destinatario de la operación de otorgar puede crear tablas en la base de datos (CREATETAB): Y = Tiene el privilegio. N = No tiene el privilegio.
BINDADDAUTH	CHAR(1)		Si el destinatario de la operación de otorgar puede crear paquetes en la base de datos (BINDADD): Y = Tiene el privilegio. N = No tiene el privilegio.
CONNECTAUTH	CHAR(1)		Si el destinatario de la operación de otorgar puede conectarse a la base de datos (CONNECT): Y = Tiene el privilegio. N = No tiene el privilegio.
NOFENCEAUTH	CHAR(1)		Si el destinatario de la operación de otorgar tiene el privilegio de crear funciones no limitadas. Y = Tiene el privilegio. N = No tiene el privilegio.
IMPLSCHEMAAUTH	CHAR(1)		Si el destinatario de la operación de otorgar puede crear esquemas implícitamente en la base de datos (IMPLICIT_SCHEMA): Y = Tiene el privilegio. N = No tiene el privilegio.
LOADAUTH	CHAR(1)		Si el destinatario de la operación de otorgar tiene autorización LOAD para la base de datos: Y = Mantiene la autoridad. N = No mantiene la autoridad.

Tabla 60. Vista de catálogo SYSCAT.DBAUTH (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
EXTERNALROUTINEAUTH	CHAR(1)		Si el destinatario de la operación de otorgar puede crear rutinas externas (CREATE_EXTERNAL_ROUTINE): Y = Mantiene la autoridad. N = No mantiene la autoridad.
QUIESCECONNECTAUTH	CHAR(1)		Si el destinatario de la operación de otorgar puede conectarse a la base de datos (QUIESCE_CONNECT): Y = Mantiene la autoridad. N = No mantiene la autoridad.

SYSCAT.DBPARTITIONGROUPDEF

Contiene una fila para cada partición que esté contenida en un grupo de particiones de base de datos.

Tabla 61. Vista de catálogo SYSCAT.DBPARTITIONGROUPDEF

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
DBPGNAME	VARCHAR(18)		El nombre del grupo de particiones de base de datos que contiene la partición de base de datos.
DBPARTITIONNUM	SMALLINT		El número de partición de una partición contenida en el grupo de particiones de base de datos. Un número de partición válido está entre 0 y 999 inclusive.
IN_USE	CHAR(1)		Estado de la partición de base de datos. A = La partición recién añadida no está en el mapa de particionamiento pero se crean los contenedores para los espacios de tablas en el grupo de particiones de base de datos. La partición se añade al mapa de particionamiento cuando se completa satisfactoriamente una operación de redistribución de grupo de particiones de base de datos. D = La partición se eliminará cuando se complete una operación de redistribución de grupo de particiones de base de datos. T = La partición que se acaba de añadir no está en el mapa de particionamiento y se ha añadido utilizando la cláusula WITHOUT TABLESPACES. Los contenedores deben añadirse específicamente a los espacios de tablas para el grupo de particiones de base de datos. Y = La partición está en el mapa de particionamiento.

SYSCAT.DBPARTITIONGROUPS

Contiene una fila para cada grupo de particiones de base de datos.

Tabla 62. Vista de catálogo SYSCAT.DBPARTITIONGROUPS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
DBPGNAME	VARCHAR(18)		Nombre del grupo de particiones de base de datos.
DEFINER	VARCHAR(128)		ID de autorización del encargado de definir el grupo de particiones de base de datos.
PMAP_ID	SMALLINT		Identificador del mapa de particionamiento en SYSCAT.PARTITIONMAPS.
REDISTRIBUTE_PMAP_ID	SMALLINT		Identificador del mapa de particionamiento que se utiliza actualmente para la redistribución. El valor es -1 si la redistribución no está en proceso actualmente.
CREATE_TIME	TIMESTAMP		Hora de creación del grupo de particiones de base de datos.
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario.

SYSCAT.EVENTMONITORS

Contiene una fila para cada supervisor de sucesos que se haya definido.

Tabla 63. Vista de catálogo SYSCAT.EVENTMONITORS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
EVMONNAME	VARCHAR(18)		Nombre del supervisor de sucesos.
DEFINER	VARCHAR(128)		ID de autorización del encargado de definir el supervisor de sucesos.
TARGET_TYPE	CHAR(1)		El tipo del destino en el que se graban los datos del suceso. Valores: F = Archivo P = Conexión T = Tabla
TARGET	VARCHAR(246)		Nombre del destino en el que se graban los datos del suceso. Nombre de vía de acceso absoluta del archivo o nombre absoluto de la conexión.
MAXFILES	INTEGER	Sí	El número máximo de archivos de sucesos que permite este supervisor de sucesos en una vía de acceso de sucesos. Nulo si no hay ningún máximo o si el tipo de destino no es FILE.
MAXFILESIZE	INTEGER	Sí	Tamaño máximo (en páginas de 4K) que cada archivo de sucesos puede alcanzar antes de que el supervisor de sucesos cree un nuevo archivo. Nulo si no hay ningún máximo o si el tipo de destino no es FILE.
BUFFERSIZE	INTEGER	Sí	Tamaño de almacenamientos intermedios (en páginas de 4K) utilizadas por los supervisores de sucesos con los destinos de archivo; de lo contrario nulo.
IO_MODE	CHAR(1)	Sí	Modalidad de E/S de archivo. B = En bloques N = No en bloques Nulo si el tipo de destino no es FILE.
WRITE_MODE	CHAR(1)	Sí	Indica cómo maneja este supervisor los datos de sucesos existentes cuando se activa el supervisor. Valores: A = Añadir R = Sustituir Nulo si el tipo de destino no es FILE.
AUTOSTART	CHAR(1)		El supervisor de sucesos se activará automáticamente cuando se inicie la base de datos. Y = Sí N = No
DBPARTITIONNUM	SMALLINT		El número de la partición de la base de datos en la que el supervisor de sucesos ejecuta y anota cronológicamente los sucesos.

Tabla 63. Vista de catálogo SYSCAT.EVENTMONITORS (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
MONSCOPE	CHAR(1)		Ámbito de supervisión: L = Local G = Global T = Por nodo en que existe un espacio de tablas Un carácter en blanco, sólo válido para supervisores de sucesos WRITE TO TABLE.
EVMON_ACTIVATES	INTEGER		El número de veces que se ha activado este monitor de sucesos.
REMARKS	VARCHAR(254)	Sí	Reservado para su utilización en el futuro.

SYSCAT.EVENTS

Contiene una fila para cada suceso que se supervisa. En general, un supervisor de sucesos supervisa varios sucesos.

Tabla 64. Vista de catálogo SYSCAT.EVENTS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
EVMONNAME	VARCHAR(18)		Nombre del supervisor de sucesos que supervisa este suceso.
TYPE	VARCHAR(18)		Tipo del suceso que se supervisa. Los valores posibles son: DATABASE CONNECTIONS TABLES STATEMENTS TRANSACTIONS DEADLOCKS DETAILDEADLOCKS TABLESPACES
FILTER	CLOB(32K)	Sí	Todo el texto de la cláusula WHERE que se aplica a este suceso.

SYSCAT.EVENTS

Contiene una fila para cada tabla de destino de un supervisor de sucesos que escribe en tablas SQL.

Tabla 65. Vista de catálogo SYSCAT.EVENTTABLES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
EVMONNAME	VARCHAR(128)		Nombre del supervisor de sucesos.
LOGICAL_GROUP	VARCHAR(18)		Nombre del grupo de datos lógicos. Puede ser uno de los siguientes: BUFFERPOOL CONN CONNHEADER CONTROL DB DEADLOCK DLCONN DLLOCK STMT SUBSECTION TABLE TABLESPACE XACT
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla de destino.
TABNAME	VARCHAR(128)		
PCTDEACTIVATE	SMALLINT		Un valor de porcentaje que especifica hasta qué punto debe estar lleno un espacio de tabla DMS para que un supervisor de sucesos lo desactive de forma automática. Establecido en 100 para los espacios de tabla SMS.

SYSCAT.FULLHIERARCHIES

Cada fila representa la relación entre una subtabla y una supertabla, un subtipo y un supertipo o una subvista y una supervista. Todas las relaciones jerárquicas, incluyendo las inmediatas, se incluyen en esta vista

Tabla 66. Vista de catálogo SYSCAT.FULLHIERARCHIES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
METATYPE	CHAR(1)		Codifica el tipo de relación: R = Entre tipos estructurados U = Entre tablas con tipo W = Entre vistas con tipo
SUB_SCHEMA	VARCHAR(128)		Nombre calificado de subtipo, subtabla o subvista.
SUB_NAME	VARCHAR(128)		
SUPER_SCHEMA	VARCHAR(128)		Nombre calificado de supertipo, supertabla o supervista.
SUPER_NAME	VARCHAR(128)		
ROOT_SCHEMA	VARCHAR(128)		Nombre calificado de la tabla, vista o tipo que esta en la raíz de la jerarquía.
ROOT_NAME	VARCHAR(128)		

SYSCAT.FUNCMAPOPTIONS

Cada fila contiene los valores de las opciones de correlación de funciones.

Tabla 67. Vista de catálogo SYSCAT.FUNCMAPOPTIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
FUNCTION_MAPPING	VARCHAR(18)		Nombre de correlación de funciones.
OPTION	VARCHAR(128)		Nombre de la opción de correlación de funciones.
SETTING	VARCHAR(255)		Valor de la opción de correlación de funciones.

SYSCAT.FUNCMAPPARMOPTIONS

Cada fila contiene los valores de las opciones de parámetro de correlación de funciones.

Tabla 68. Vista de catálogo SYSCAT.FUNCMAPPARMOPTIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
FUNCTION_MAPPING	VARCHAR(18)		Nombre de correlación de funciones.
ORDINAL	SMALLINT		Posición de parámetro
LOCATION	CHAR(1)		L = Local R = Remoto
OPTION	VARCHAR(128)		Nombre de la opción de parámetro de correlación de funciones.
SETTING	VARCHAR(255)		Valor de la opción de parámetro de correlación de funciones.

SYSCAT.FUNCMAPPINGS

Cada fila contiene las correlaciones de funciones.

Tabla 69. Vista de catálogo SYSCAT.FUNCMAPPINGS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
FUNCTION_ MAPPING	VARCHAR(128)		Nombre de correlación de funciones (puede ser generado por el sistema).
FUNCSCHEMA	VARCHAR(128)	Sí	Esquema de función. Nulo si es una función incorporada del sistema.
FUNCNAME	VARCHAR(1024)	Sí	Nombre de la función local (incorporada o definida por el usuario).
FUNCID	INTEGER	Sí	Identificador asignado internamente.
SPECIFICNAME	VARCHAR(128)	Sí	Nombre de la instancia de función local.
DEFINER	VARCHAR(128)		ID de autorización bajo el cual se ha creado esta correlación.
WRAPNAME	VARCHAR(128)	Sí	Nombre de reiniciador al que se aplica esta correlación.
SERVERNAME	VARCHAR(128)	Sí	Nombre de la fuente de datos.
SERVERTYPE	VARCHAR (30)	Sí	Tipo de fuente de datos a la que se aplica la correlación.
SERVERVERSION	VARCHAR(18)	Sí	Versión del tipo de servidor al que se aplica la correlación.
CREATE_TIME	TIMESTAMP	Sí	Hora en que se ha creado la correlación.
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario o nulo.

SYSCAT.HIERARCHIES

Cada fila representa la relación entre una subtabla y su supertabla inmediata, un subtipo y su supertipo inmediato o una subvista y su supervista inmediata. Sólo las relaciones jerárquicas inmediatas se incluyen en esta vista.

Tabla 70. Vista de catálogo SYSCAT.HIERARCHIES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
METATYPE	CHAR(1)		Codifica el tipo de relación: R = Entre tipos estructurados U = Entre tablas con tipo W = Entre vistas con tipo
SUB_SCHEMA	VARCHAR(128)		Nombre calificado de subtipo, subtabla o subvista.
SUB_NAME	VARCHAR(128)		
SUPER_SCHEMA	VARCHAR(128)		Nombre calificado de supertipo, supertabla o supervista.
SUPER_NAME	VARCHAR(128)		
ROOT_SCHEMA	VARCHAR(128)		Nombre calificado de la tabla, vista o tipo que esta en la raíz de la jerarquía.
ROOT_NAME	VARCHAR(128)		

SYSCAT.INDEXAUTH

Contiene una fila para cada privilegio que se posea en un índice.

Tabla 71. Vista de catálogo SYSCAT.INDEXAUTH

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
GRANTOR	VARCHAR(128)		ID de autorización del usuario que ha otorgado los privilegios.
GRANTEE	VARCHAR(128)		ID de autorización del usuario o grupo que tiene los privilegios.
GRANTEETYPE	CHAR(1)		U = Se otorga a un usuario individual. G = Se otorga a un grupo.
INDSCHEMA	VARCHAR(128)		Nombre calificado del índice.
INDNAME	VARCHAR(18)		
CONTROLAUTH	CHAR(1)		Si el destinatario de la operación de otorgar posee el privilegio CONTROL en el índice: Y = Tiene el privilegio. N = No tiene el privilegio.

SYSCAT.INDEXCOLUSE

Lista todas las columnas que participan en un índice.

Tabla 72. Vista de catálogo SYSCAT.INDEXCOLUSE

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
INDSCHEMA	VARCHAR(128)		Nombre calificado del índice.
INDNAME	VARCHAR(18)		
COLNAME	VARCHAR(128)		Nombre de la columna.
COLSEQ	SMALLINT		Posición numérica de la columna en el índice (posición inicial = 1).
COLORDER	CHAR(1)		Orden de los valores de esta columna en el índice. Valores: A = Ascendente D = Descendente I = Columna INCLUDE (se pasa por alto el orden)

SYSCAT.INDEXDEP

Cada fila representa una dependencia de un índice respecto de algún otro objeto.

Tabla 73. Vista de catálogo SYSCAT.INDEXDEP

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
INDSCHEMA	VARCHAR(128)		Nombre calificado del índice que depende de otro objeto.
INDNAME	VARCHAR(18)		
BTYPE	CHAR(1)		Tipo de objeto del que depende el índice. A = Seudónimo F = Instancia de función O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o vistas R = Tipo estructurado S = Tabla de consulta materializada T = Tabla U = Tabla con tipo V = Vista W = Vista con tipo X = Extensión de índice
BSCHEMA	VARCHAR(128)		Nombre calificado del objeto del que depende el índice.
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Sí	Si BTYPE= O, S, T, U, V ó W, codifica los privilegios en la tabla o vista que necesite el índice dependiente; de lo contrario, es nulo.

SYSCAT.INDEXES

Contiene una fila para cada índice (incluidos los índices heredados donde sea aplicable) que se define para una tabla.

Tabla 74. Vista de catálogo SYSCAT.INDEXES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
INDSCHEMA	VARCHAR(128)		Nombre del índice.
INDNAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		Usuario que ha creado el índice.
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla o apodo en el que se define el índice.
TABNAME	VARCHAR(128)		
COLNAMES	VARCHAR(640)		Lista de nombres de columna, cada uno precedido por un signo + o − para indicar orden ascendente o descendente respectivamente. Aviso: Esta columna se eliminará en el futuro. Utilice SYSCAT.INDEXCOLUSE para obtener esta información.
UNIQUERULE	CHAR(1)		Regla de unicidad: D = Los duplicados están permitidos P = Índice primario U = Sólo se permiten entradas exclusivas
MADE_UNIQUE	CHAR(1)		Y = El índice era de no unicidad originalmente, pero se ha convertido en un índice de unicidad para dar soporte a una restricción de clave primaria o de unicidad. Si se elimina la restricción, el índice volverá a ser de no unicidad. N = El índice permanece tal como se ha creado.
COLCOUNT	SMALLINT		El número de columnas de la clave, más el número de columnas INCLUDE, si hay alguna.
UNIQUE_COLCOUNT	SMALLINT		El número de columnas necesarias para una clave de unicidad. Siempre <=COLCOUNT. < COLCOUNT solamente si hay columnas INCLUDE. –1 si el índice no tiene clave de unicidad (permite los duplicados).
INDEXTYPE	CHAR(4)		Tipo de índice. CLUS = Clúster REG = Normal DIM = Índice de bloques de dimensión BLOK = Índice de bloques

Tabla 74. Vista de catálogo SYSCAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
ENTRYTYPE	CHAR(1)		H = Un índice en una tabla de jerarquía (tabla-H) L = Índice lógico en una tabla con tipo en blanco si es un índice en una tabla de no tipo
PCTFREE	SMALLINT		Porcentaje de cada página de hoja de índice que se va a reservar durante la creación inicial del índice. Este espacio está disponible para inserciones futuras después de la creación del índice.
IID	SMALLINT		ID interno de índice.
NLEAF	INTEGER		Número de páginas de índice; –1 si no se reúnen estadísticas.
NLEVELS	SMALLINT		Número de niveles de índice; –1 si no se reúnen estadísticas.
FIRSTKEYCARD	BIGINT		El número de valores de primera clave diferenciada; –1 si no se reúnen estadísticas.
FIRST2KEYCARD	BIGINT		El número de claves diferenciadas que utilizan las dos primeras columnas del índice; –1 si no hay datos estadísticos o no son aplicables.
FIRST3KEYCARD	BIGINT		El número de claves diferenciadas que utilizan las tres primeras columnas del índice; –1 si no hay datos estadísticos o no son aplicables.
FIRST4KEYCARD	BIGINT		El número de claves diferenciadas que utilizan las cuatro primeras columnas del índice; –1 si no hay datos estadísticos o no son aplicables.
FULLKEYCARD	BIGINT		El número de valores de clave completa diferenciada; –1 si no se reúnen estadísticas.
CLUSTERRATIO	SMALLINT		Nivel de clúster de los datos con el índice; –1 si no se reúnen estadísticas o si se reúnen estadísticas de índice detalladas (en este caso, se utilizará CLUSTERFACTOR en su lugar).
CLUSTERFACTOR	DOUBLE		Medición más refinada del nivel de clúster o –1 si no se reúnen estadísticas de índice detalladas o si el índice está definido en un apodo.
SEQUENTIAL_PAGES	INTEGER		El número de páginas ubicadas en disco por orden de clave de índice, con pocos o ningún hueco entre ellas; –1 si no hay datos estadísticos disponibles.

Tabla 74. Vista de catálogo SYSCAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
DENSITY	INTEGER		Proporción de SEQUENTIAL_PAGES para numerar las páginas del rango de páginas ocupadas por el índice, expresada como un porcentaje (entero entre 0 y 100; –1 si no hay datos estadísticos disponibles.)
USER_DEFINED	SMALLINT		1 si un usuario ha definido este índice y no se ha eliminado; de lo contrario, 0.
SYSTEM_REQUIRED	SMALLINT		Los valores válidos son: 1 si se cumple una de las condiciones siguientes: – Este índice es necesario para una restricción de clave primaria o de unicidad o bien este índice es un índice de bloques de dimensión o un índice de bloques compuestos para una tabla con clústeres de varias dimensiones (MDC). – Se trata de un índice en la columna OID de una tabla con tipo. 2 si se cumplen estas dos condiciones: – Este índice es necesario para una restricción de clave primaria o de unicidad o bien este índice es un índice de bloques de dimensión o un índice de bloques compuestos para una tabla MDC. – Se trata de un índice en la columna OID de una tabla con tipo. De lo contrario, 0.
CREATE_TIME	TIMESTAMP		Hora en que se ha creado el índice.
STATS_TIME	TIMESTAMP	Sí	Última vez que se ha realizado un cambio en las estadísticas registradas para este índice. Nulo si no hay estadísticas disponibles.
PAGE_FETCH_PAIRS	VARCHAR(254)		Una lista de pares de enteros, representada en la forma de caracteres. Cada par representa el número de páginas de un almacenamiento intermedio hipotético y el número de lecturas de páginas necesario para explorar la tabla con este índice utilizando dicho almacenamiento intermedio hipotético. (Serie de longitud cero si no hay datos disponibles.)
MINPCTUSED	SMALLINT		Si no es cero, se habilita la desfragmentación del índice en línea y el valor es el umbral del espacio mínimo utilizado antes de fusionar las páginas.
REVERSE_SCANS	CHAR(1)		Y = El índice soporta exploraciones invertidas N = El índice no soporta exploraciones invertidas

Tabla 74. Vista de catálogo SYSCAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
INTERNAL_FORMAT	SMALLINT		Los valores válidos son: 1 si el índice no tiene punteros que apunten hacia atrás >= 2 si el índice sí que tiene punteros que apunten hacia atrás 6 si se trata de un índice de bloques compuestos
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario o nulo.
IESCHEMA	VARCHAR(128)	Sí	Nombre calificado de la extensión de índice. Nulo para índices ordinarios.
IENAME	VARCHAR(18)	Sí	
IEARGUMENTS	CLOB(32K)	Sí	Información externa del parámetro especificado cuando se crea el índice. Nulo para índices ordinarios.
INDEX_OBJECTID	INTEGER		Identificador del objeto de índice de la tabla.
NUMRIDS	BIGINT		Número total de identificadores de filas (RID) del índice.
NUMRIDS_DELETED	BIGINT		Número total de identificadores de filas del índice que están marcados como suprimidos, excluidos los identificadores de filas de páginas en las que todos los identificadores de filas están marcados como suprimidos.
NUM_EMPTY_LEAFS	BIGINT		Número total de páginas del índice que tienen todos los identificadores de filas marcados como suprimidos.
AVERAGE_RANDOM_ FETCH_PAGES	DOUBLE		Promedio de páginas de la tabla aleatorias entre los accesos a páginas secuenciales al captar utilizando el índice; –1 si no se conoce. Consulte las notas 1 y 2.
AVERAGE_RANDOM_ PAGES	DOUBLE		Promedio de páginas del índice aleatorias entre los accesos a páginas del índice; –1 si no se conoce. Consulte la nota 2.
AVERAGE_SEQUENCE_ GAP	DOUBLE		Espacio entre las secuencias de páginas del índice. Detectado mediante una exploración de las páginas del índice, cada espacio representa el promedio de páginas del índice que deben captarse de forma aleatoria entre las secuencias de las páginas del índice; –1 si no se conoce. Consulte la nota 2.

Tabla 74. Vista de catálogo SYSCAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
AVERAGE_SEQUENCE_ FETCH_GAP	DOUBLE		Espacio entre las secuencias de páginas de la tabla al captar utilizando el índice. Detectado mediante una exploración de las páginas del índice, cada espacio representa el promedio de páginas de la tabla que deben captarse de forma aleatoria entre las secuencias de las páginas de la tabla; –1 si no se conoce. Consulte las notas 1 y 2.
AVERAGE_SEQUENCE_ PAGES	DOUBLE		Promedio de páginas del índice accesibles en secuencia (es decir, el número de páginas del índice que la captación previa detectaría que forman una secuencia); –1 si no se conoce. Consulte la nota 2.
AVERAGE_SEQUENCE_ FETCH_PAGES	DOUBLE		Promedio de páginas de la tabla accesibles en secuencia (es decir, el número de páginas de la tabla que la captación previa detectaría que forman una secuencia) al captar utilizando el índice; –1 si no se conoce. Consulte las notas 1 y 2.
TBSPACEID	INTEGER		Identificador interno del espacio de tabla del índice.

Notas:

1. Cuando se utilizan espacios de tabla DMS, no puede calcularse esta estadística.
2. No se recopilan estadísticas de captación previa durante una operación LOAD...STATISTICS YES o CREATE INDEX...COLLECT STATISTICS ni cuando el parámetro de configuración *seqdetect* está desactivado.

Información relacionada:

• "SYSCAT.INDEXCOLUSE" en la página 592

SYSCAT.INDEXEXPLOITRULES

Cada fila representa una explotación de índice.

Tabla 75. Vista de catálogo SYSCAT.INDEXEXPLOITRULES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
FUNCID	INTEGER		ID de función.
SPECID	SMALLINT		El número de la especificación de predicado dentro de la sentencia CREATE FUNCTION.
IESCHEMA	VARCHAR(128)		Nombre calificado de la extensión del índice.
IENAME	VARCHAR(18)		
RULEID	SMALLINT		ID exclusivo de la regla de explotación.
SEARCHMETHODID	SMALLINT		ID del método de búsqueda en la extensión de índice específica.
SEARCHKEY	VARCHAR(320)		Clave utilizada para explotar el índice.
SEARCHARGUMENT	VARCHAR(1800)		Argumentos de búsqueda utilizados en la explotación de índice.
EXACT	CHAR(1)		Indica si la búsqueda en el índice es o no es exacta respecto a la evaluación del predicado. Los valores posibles son: Y = La búsqueda en el índice es exacta. N = La búsqueda en el índice no es exacta.

SYSCAT.INDEXEXTENSIONDEP

Contiene una fila para cada dependencia que las extensiones de índice tienen sobre diversos objetos de base de datos.

Tabla 76. Vista de catálogo SYSCAT.INDEXEXTENSIONDEP

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
IESCHEMA	VARCHAR(128)		Nombre calificado de la extensión de índice que depende de otro objeto.
IENAME	VARCHAR(18)		
BTYPE	CHAR(1)		Tipo de objeto del que depende la extensión de índice: A = Seudónimo F = Instancia de función o instancia de método J = Definición de servidor O = Dependencia "externa" sobre privilegio SELECT jerárquico R = Tipo estructurado S = Tabla de consulta materializada T = Tabla (sin tipo) U = Tabla con tipo V = Vista (sin tipo) W = Vista con tipo X = Extensión de índice
BSCHEMA	VARCHAR(128)		Nombre calificado del objeto del que depende la extensión de índice. (Si BTYPE='F', se trata del nombre específico de una función).
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Sí	Si BTYPE='O', 'T', 'U', 'V' ó 'W', codifica los privilegios que un activador dependiente necesita en la tabla (o vista); en caso contrario su valor es nulo.

SYSCAT.INDEXEXTENSIONMETHODS

Cada fila representa un método de búsqueda. Una extensión de índice puede contener varios métodos de búsqueda.

Tabla 77. Vista de catálogo SYSCAT.INDEXEXTENSIONMETHODS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
METHODNAME	VARCHAR(18)		Nombre del método de búsqueda.
METHODID	SMALLINT		El número del método en la extensión de índice.
IESCHEMA	VARCHAR(128)		Nombre calificado de la extensión de índice.
IENAME	VARCHAR(18)		
RANGEFUNCSCHEMA	VARCHAR(128)		Nombre calificado de la función de rangos.
RANGEFUNCNAME	VARCHAR(18)		
RANGESPECIFICNAME	VARCHAR(18)		Nombre específico de la función de rangos.
FILTERFUNCSCHEMA	VARCHAR(128)		Nombre calificado de la función de filtro.
FILTERFUNCNAME	VARCHAR(18)		
FILTERSPECIFICNAME	VARCHAR(18)		Nombre específico de la función de filtro.
REMARKS	VARCHAR(254)	Sí	Proporcionado por el usuario o nulo.

SYSCAT.INDEXEXTENSIONPARMS

Cada fila representa un parámetro de instancia de la extensión de índice o una definición de la clave fuente.

Tabla 78. Vista de catálogo SYSCAT.INDEXEXTENSIONPARMS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
IESCHEMA	VARCHAR(128)		Nombre calificado de la extensión de índice.
IENAME	VARCHAR(18)		
ORDINAL	SMALLINT		El número de orden del parámetro o clave fuente.
PARMNAME	VARCHAR(18)		Nombre del parámetro o clave fuente.
TYPESCHEMA	VARCHAR(128)		Nombre calificado del parámetro de instancia o del tipo de datos de la clave fuente.
TYPENAME	VARCHAR(18)		
LENGTH	INTEGER		Longitud del parámetro de instancia o del tipo de datos de la clave fuente.
SCALE	SMALLINT		Escala del parámetro de instancia o del tipo de datos de la clave fuente. Igual a 0 si no es aplicable.
PARMTYPE	CHAR(1)		Tipo representado por la fila: P = parámetro de extensión de índice K = columna de clave
CODEPAGE	SMALLINT		Página de códigos del parámetro de extensión de índice. Es igual a 0 si no es un tipo de serie de caracteres.

SYSCAT.INDEXEXTENSIONS

Contiene una fila para cada extensión de índice.

Tabla 79. Vista de catálogo SYSCAT.INDEXEXTENSIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
IESCHEMA	VARCHAR(128)		Nombre calificado de la extensión de índice.
IENAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		ID de autorización con el cual se ha definido la extensión de índice.
CREATE_TIME	TIMESTAMP		Hora en la que se definió la extensión de índice.
KEYGENFUNCSCHEMA	VARCHAR(128)		Nombre calificado de la función de generación de claves.
KEYGENFUNCNAME	VARCHAR(18)		
KEYGENSPECIFICNAME	VARCHAR(18)		Nombre específico de la función de generación de claves.
TEXT	CLOB(64K)		El texto completo de la sentencia CREATE INDEX EXTENSION.
REMARKS	VARCHAR(254)		Comentario suministrado por el usuario o nulo.

SYSCAT.INDEXOPTIONS

Cada fila contiene los valores de las opciones específicas de índice.

Tabla 80. Vista de catálogo SYSCAT.INDEXOPTIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
INDSCHEMA	VARCHAR(128)		Nombre de esquema del índice.
INDNAME	VARCHAR(18)		Nombre local del índice.
OPTION	VARCHAR(128)		Nombre de la opción de índice.
SETTING	VARCHAR(255)		Valor.

SYSCAT.KEYCOLUSE

Lista todas las columnas que participan en una clave (incluidas las claves primarias o de unicidad heredadas donde sea aplicable) definidas por una restricción de unicidad, de clave primaria o de clave foránea.

Tabla 81. Vista de catálogo SYSCAT.KEYCOLUSE

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
CONSTNAME	VARCHAR(18)		Nombre de la restricción (exclusivo en una tabla).
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla que contiene la columna.
TABNAME	VARCHAR(128)		
COLNAME	VARCHAR(128)		Nombre de la columna.
COLSEQ	SMALLINT		Posición numérica de la columna en la clave (posición inicial=1).

SYSCAT.NAMEMAPPINGS

Cada fila representa la correlación entre los objetos lógicos y los objetos de implantación correspondientes que implantan los objetos lógicos.

Tabla 82. Vista de catálogo SYSCAT.NAMEMAPPINGS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TYPE	CHAR(1)		C = Columna I = Índice U = Tabla con tipo
LOGICAL_SCHEMA	VARCHAR(128)		Nombre calificado del objeto lógico.
LOGICAL_NAME	VARCHAR(128)		
LOGICAL_COLNAME	VARCHAR(128)	Sí	Si TYPE = C, el nombre de la columna lógica. De lo contrario es nulo.
IMPL_SCHEMA	VARCHAR(128)		Nombre calificado del objeto de implantación que implanta el objeto lógico.
IMPL_NAME	VARCHAR(128)		
IMPL_COLNAME	VARCHAR(128)	Sí	Si TYPE = C, el nombre de la columna de implantación. De lo contrario es nulo.

SYSCAT.PACKAGEAUTH

Contiene una fila para cada privilegio que se posea en un paquete.

Tabla 83. Vista de catálogo SYSCAT.PACKAGEAUTH

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
GRANTOR	VARCHAR(128)		ID de autorización del usuario que ha otorgado los privilegios.
GRANTEE	VARCHAR(128)		ID de autorización del usuario o grupo que tiene los privilegios.
GRANTEETYPE	CHAR(1)		U = Se otorga a un usuario individual. G = Se otorga a un grupo.
PKGSCHEMA	VARCHAR(128)		Nombre del paquete en el que se tienen los privilegios.
PKGNAME	CHAR(8)		
CONTROLAUTH	CHAR(1)		Indica si el destinatario de la operación de otorgar posee el privilegio CONTROL en el paquete: Y = Tiene el privilegio. N = No tiene el privilegio.
BINDAUTH	CHAR(1)		Indica si el destinatario de la operación de otorgar posee el privilegio BIND en el paquete: Y = Tiene el privilegio. N = No tiene el privilegio. G = Tiene el privilegio y es otorgable.
EXECUTEAUTH	CHAR(1)		Indica si el destinatario de la operación de otorgar posee el privilegio EXECUTE en el paquete: Y = Tiene el privilegio. N = No tiene el privilegio. G = Tiene el privilegio y es otorgable.

SYSCAT.PACKAGEDEP

Contiene una fila para cada dependencia que los paquetes tienen sobre índices, tablas, vistas, activadores, funciones, seudónimos, tipos y jerarquías.

Tabla 84. Vista de catálogo SYSCAT.PACKAGEDEP

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
PKGSCHEMA	VARCHAR(128)		Nombre del paquete.
PKGNAME	CHAR(8)		
UNIQUEID	CHAR(8)		Información de fecha y hora interna que indica cuándo se creó el paquete por primera vez. Útil para identificar un paquete concreto cuando existen varios paquetes con el mismo nombre.
PKGVERSION	VARCHAR(64)		Identificador de versión del paquete.
BINDER	VARCHAR(128)	Sí	Vinculador del paquete.
BTYPE	CHAR(1)		Tipo de objeto BNAME: A = Seudónimo B = Activador D = Definición de servidor F = Instancia de función I = Índice M = Correlación de funciones N = Apodo O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas P = Tamaño de página R = Tipo estructurado S = Tabla de consulta materializada T = Tabla U = Tabla con tipo V = Vista W = Vista con tipo
BSCHEMA	VARCHAR(128)		Nombre calificado de un objeto del que depende el paquete.
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Sí	Si BTYPE es O, S, T, U, V o W cuando codifica los privilegios que necesita este paquete (Select, Insert, Delete, Update).

Nota: Si se descarta una instancia de función de la que depende un paquete, éste se coloca en un estado "no operativo" y debe volver a vincularse explícitamente. Si se descarta cualquier otro objeto del que depende un paquete, éste se coloca en un estado "no válido" y el sistema intentará vincularlo de forma automática cuando se haga referencia al mismo por primera vez.

SYSCAT.PACKAGES

Contiene una fila para cada paquete que se ha creado mediante la vinculación de un programa de aplicación.

Tabla 85. Vista de catálogo SYSCAT.PACKAGES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
PKGSCHEMA	VARCHAR(128)		Calificador de esquema del nombre del paquete.
PKGNAME	CHAR(8)		Identificador no calificado del nombre del paquete
PKGVERSION	VARCHAR(64)		Identificador de versión del nombre del paquete.
BOUNDBY	VARCHAR(128)		ID de autorización (OWNER) del vinculador del paquete.
DEFINER	VARCHAR(128)		ID de usuario bajo el cual se ha vinculado el paquete.
DEFAULT_SCHEMA	VARCHAR(128)		Nombre de esquema por omisión (QUALIFIER) utilizado para nombres no calificados en sentencias de SQL estático.
VALID	CHAR(1)		Y = Válido N = No válido X = El paquete no es operativo porque alguna instancia de función de la que depende se ha desactivado. Se necesita volver a ejecutar bind. (Si se descarta una instancia de función de la que depende un paquete, éste se coloca en un estado "no operativo" y debe volver a vincularse explícitamente. Si se descarta cualquier otro objeto del que depende un paquete, éste se coloca en un estado "no válido" y el sistema intentará vincularlo de forma automática cuando se haga referencia al mismo por primera vez).
UNIQUE_ID	CHAR(8)		Información de fecha y hora interna que indica cuándo se creó el paquete por primera vez. Útil para identificar un paquete concreto cuando existen varios paquetes con el mismo nombre.
TOTAL_SECT	SMALLINT		El número total de secciones en el paquete.
FORMAT	CHAR(1)		Formato de fecha y hora asociado con el paquete: 0 = Formato asociado con el código territorial del cliente 1 = Fecha y hora USA 2 = Fecha EUR, hora EUR 3 = Fecha ISO, hora ISO 4 = Fecha JIS, hora JIS 5 = Fecha LOCAL, hora LOCAL
ISOLATION	CHAR(2)	Sí	Nivel de aislamiento: RR = Lectura repetible RS = Estabilidad de lectura CS = Estabilidad del cursor UR = Lectura no confirmada

Tabla 85. Vista de catálogo SYSCAT.PACKAGES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
BLOCKING	CHAR(1)	Sí	Opción de bloqueo del cursor: N = Sin bloqueo U = Bloqueo de cursores no ambiguos B = Bloqueo de todos los cursores
INSERT_BUF	CHAR(1)		Opción de inserción utilizada durante la vinculación: Y = Las inserciones se ponen en almacenamiento intermedio N = Las inserciones no se ponen en almacenamiento intermedio
REOPTVAR	CHAR(1)		Indica si la vía de acceso se reoptimiza durante la ejecución utilizando los valores de variables de entrada siguientes: A = La vía de acceso se reoptimiza durante cada petición OPEN o EXECUTE. N = La vía de acceso no se reoptimiza durante la ejecución. O = La vía de acceso sólo se reoptimiza durante la primera petición OPEN o EXECUTE. Con posterioridad, se guarda en la antememoria.
OS_PTR_SIZE	INTEGER		Indica el tamaño de las palabras para la plataforma en la que se ha creado el paquete: 32 = El paquete es un paquete de 32 bits. 64 = El paquete es un paquete de 64 bits.
LANG_LEVEL	CHAR(1)	Sí	Valor LANGLEVEL utilizado durante BIND: 0 = SAA1 1 = MIA 2 = SQL92E
FUNC_PATH	VARCHAR(254)		La vía de acceso de SQL utilizada por el último mandato BIND para este paquete. Se utiliza como la vía de acceso por omisión para REBIND. SYSIBM para los paquetes anteriores a la Versión 2.
QUERYOPT	INTEGER		La clase de optimización bajo la que se ha vinculado este paquete. Utilizado para volver a vincular (REBIND). Las clases son: 0, 1, 3, 5 y 9.
EXPLAIN_LEVEL	CHAR(1)		Indica si se ha invocado Explain utilizando la opción de vinculación EXPLAIN o EXPLSNAP. P = Nivel de selección de planificación En blanco si 'No' se invoca Explain
EXPLAIN_MODE	CHAR(1)		Valor de la opción de vinculación EXPLAIN: Y = Sí (estática) N = No A = Todos (estática y dinámica)

Tabla 85. Vista de catálogo SYSCAT.PACKAGES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
EXPLAIN_SNAPSHOT	CHAR(1)		Valor de la opción de vinculación EXPLSNAP: Y = Sí (estática) N = No A = Todos (estática y dinámica)
SQLWARN	CHAR(1)		¿Se devuelven a la aplicación los SQLCODE positivos resultantes de las sentencias de SQL dinámico? Y = Sí N = No, se suprimen
SQLMATHWARN	CHAR(1)		Valor del parámetro de configuración de base de datos DFT_SQLMATHWARN en el tiempo de vinculación. ¿Se tratan los errores aritméticos y los errores de conversión de recuperación de las sentencias de SQL estático como nulos con un aviso? Y = Sí N = No, se suprimen
EXPLICIT_BIND_TIME	TIMESTAMP		La hora en la que este paquete se ha vinculado o vuelto a vincular explícitamente por última vez. Cuando el paquete se vuelva a vincular explícitamente, no se seleccionará ninguna instancia de función que se haya creado con posterioridad a esta hora.
LAST_BIND_TIME	TIMESTAMP		La hora en la que el paquete se ha vinculado o vuelto a vincular explícita o implícitamente por última vez.
CODEPAGE	SMALLINT		Página de códigos de la aplicación en tiempo de vinculación (-1 si no se conoce).
DEGREE	CHAR(5)		Indica el límite en el paralelismo intrapartición (como opción de vinculación) cuando se ha vinculado el paquete. 1 = Sin paralelismo intrapartición. 2 32 767 = Nivel de paralelismo intrapartición. ANY = El nivel lo ha determinado el gestor de bases de datos.
MULTINODE_PLANS	CHAR(1)		Y = El paquete se ha vinculado en un entorno de múltiples particiones. N = El paquete se ha vinculado en un entorno de una sola partición.

Tabla 85. Vista de catálogo SYSCAT.PACKAGES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
INTRA_PARALLEL	CHAR(1)		Indica la utilización del paralelismo intrapartición por las sentencias de SQL estático dentro del paquete. Y = una o varias sentencias de SQL estático del paquete utilizan paralelismo intrapartición. N = ninguna sentencia de SQL estático del paquete utiliza paralelismo intrapartición. F = una o varias sentencias de SQL estático del paquete utilizan paralelismo intrapartición; se ha inhabilitado este paralelismo para su uso en un sistema que no está configurado para paralelismo intrapartición.
VALIDATE	CHAR(1)		B = Toda comprobación debe realizarse durante BIND R = Reservado
DYNAMICRULES	CHAR(1)		B = BIND. Las sentencias de SQL dinámico se ejecutan con comportamiento de vinculación. D = DEFINEBIND. Cuando el paquete se ejecuta en el contexto de una rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento de definición. Cuando el paquete no se ejecuta en el contexto de una rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento de vinculación. E = DEFINERUN. Cuando el paquete se ejecuta en el contexto de una rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento de definición. Cuando el paquete no se ejecuta en el contexto de una rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento de ejecución. H = INVOKEBIND. Cuando el paquete se ejecuta en el contexto de una rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento de invocación. Cuando el paquete no se ejecuta en el contexto de una rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento de vinculación. I = INVOKERUN. Cuando el paquete se ejecuta en el contexto de una rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento de invocación. Cuando el paquete no se ejecuta en el contexto de una rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento de ejecución. R = RUN. Las sentencias de SQL dinámico se ejecutan con comportamiento de ejecución. Éste es el valor por omisión.

Tabla 85. Vista de catálogo SYSCAT.PACKAGES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
SQLERROR	CHAR(1)		Indica la opción SQLERROR en el submandato más reciente que ha vinculado o revinculado el paquete. C = Reservado N = Sin paquete
REFRESHAGE	DECIMAL (20.6)		Duración de la indicación de fecha y hora indicando la longitud máxima del tiempo entre cuando una sentencia REFRESH TABLE se ejecuta para una tabla de consulta materializada y cuando la tabla de consulta materializada se utiliza en lugar de una tabla base.
TRANSFORMGROUP	CHAR(1024)	Sí	Serie que contiene la opción de vinculación del grupo de transformación
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario o nulo.

SYSCAT.PARTITIONMAPS

Contiene una fila para cada mapa de particionamiento que se utiliza para distribuir las filas de tablas entre las particiones de un grupo de particiones, basándose en la clave de particionamiento de tabla.

Tabla 86. Vista de catálogo SYSCAT.PARTITIONMAPS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
PMAP_ID	SMALLINT		Identificador del mapa de particionamiento.
PARTITIONMAP	LONG VARCHAR FOR BIT DATA		El mapa de particionamiento real, un vector de 4 096 enteros de dos bytes para un grupo de particiones de base de datos de múltiples particiones. Para un grupo de particiones de base de datos de una sola partición, hay una entrada que indica el número de partición de la partición individual.

SYSCAT.PASSTHRUAUTH

Esta vista de catálogo contiene información acerca de las autorizaciones para consultar fuentes de datos en sesiones de paso a través. Una restricción en la tabla base necesita que los valores de SERVER se correspondan con los valores de la columna SERVER de SYSCAT.SERVERS. Ningún campo de SYSCAT.PASSTHRUAUTH puede ser nulo.

Tabla 87. Columnas de la vista de catálogo SYSCAT.PASSTHRUAUTH

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
GRANTOR	VARCHAR(128)		ID de autorización del usuario que ha otorgado el privilegio.
GRANTEE	VARCHAR(128)		ID de autorización del usuario o del grupo que tiene el privilegio.
GRANTEETYPE	CHAR(1)		Una letra que especifica el tipo al que se ha otorgado: U = Se otorga a un usuario individual. G = Se otorga a un grupo.
SERVERNAME	VARCHAR(128)		Nombre de la fuente de datos para la que se está otorgando la autorización al usuario o al grupo.

SYSCAT.PREDICATESPECS

Cada fila representa una especificación del predicado.

Tabla 88. Vista de catálogo SYSCAT.PREDICATESPECS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
FUNCSCHEMA	VARCHAR(128)		Nombre calificado de la función.
FUNCNAME	VARCHAR(128)		
SPECIFICNAME	VARCHAR(128)		El nombre de la instancia de función.
FUNCID	INTEGER		ID de función.
SPECID	SMALLINT		ID de la especificación de predicado.
CONTEXTOP	CHAR(8)		El operador de comparación es uno de los operadores relacionales internos (=, <, >=, etc.)
CONTEXTEXP	CLOB(32K)		Constante o una expresión SQL.
FILTERTEXT	CLOB(32K)	Sí	Texto de la expresión de filtro de datos.

SYSCAT.PROCOPTIONS

Cada fila contiene los valores de las opciones específicas de servidor

Tabla 89. Vista de catálogo SYSCAT.PROCOPTIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
PROCSCHEMA	VARCHAR(128)		Calificador para el nombre o apodo del procedimiento almacenado.
PROCNAME	VARCHAR(128)		Nombre o apodo del procedimiento almacenado.
OPTION	VARCHAR(128)		Nombre de la opción de procedimiento almacenado.
SETTING	VARCHAR(255)		Valor de la opción de procedimiento almacenado.

SYSCAT.PROCPARMOPTIONS

Cada fila contiene los valores de las opciones específicas del parámetro de procedimiento.

Tabla 90. Vista de catálogo SYSCAT.PROCPARMOPTIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
PROCSCHEMA	VARCHAR(128)		Nombre o apodo de procedimiento calificado.
PROCNAME	VARCHAR(128)		
ORDINAL	SMALLINT		Posición numérica del parámetro dentro de la signatura del procedimiento.
OPTION	VARCHAR(128)		Nombre de la opción de parámetro del procedimiento almacenado.
SETTING	VARCHAR(255)		Valor de la opción de parámetro del procedimiento almacenado.

SYSCAT.REFERENCES

Contiene una fila para cada restricción de referencia definida.

Tabla 91. Vista de catálogo SYSCAT.REFERENCES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
CONSTNAME	VARCHAR(18)		Nombre de la restricción.
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla.
TABNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		Usuario que ha creado la restricción.
REFKEYNAME	VARCHAR(18)		Nombre de la clave padre.
REFTABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla padre.
REFTABNAME	VARCHAR(128)		
COLCOUNT	SMALLINT		El número de columnas de la clave foránea.
DELETERULE	CHAR(1)		Regla de supresión. A = NO ACTION C = CASCADE N = SET NULL R = RESTRICT
UPDATERULE	CHAR(1)		Regla de actualización: A = NO ACTION R = RESTRICT
CREATE_TIME	TIMESTAMP		La indicación de fecha y hora cuando se ha definido la restricción de referencia.
FK_COLNAMES	VARCHAR (640)		Lista de nombres de columna de clave foránea. Aviso: Esta columna se eliminará en el futuro. Utilice SYSCAT.KEYCOLUSE para obtener esta información.
PK_COLNAMES	VARCHAR (640)		Lista de nombres de columna de clave padre. Aviso: Esta columna se eliminará en el futuro. Utilice SYSCAT.KEYCOLUSE para obtener esta información.

Nota: La vista SYSCAT.REFERENCES se basa en la tabla SYSIBM.SYSRELS de la Versión 1.

Información relacionada:

- "SYSCAT.KEYCOLUSE" en la página 605

SYSCAT.REVTYPEMAPPINGS

Cada fila contiene las correlaciones de tipos de datos invertidos (correlaciones de tipos de datos definidos localmente para los tipos de datos de fuente de datos). No hay datos en esta versión. Se define para una posible utilización futura con las correlaciones de tipos de datos.

Tabla 92. Vista de catálogo SYSCAT.REVTYPEMAPPINGS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TYPE_MAPPING	VARCHAR(18)		Nombre de la correlación de tipos invertidos (puede ser generado por el sistema).
TYPESCHEMA	VARCHAR(128)	Sí	Nombre de esquema del tipo. Nulo para tipos internos del sistema.
TYPENAME	VARCHAR(18)		Nombre del tipo local de una correlación de tipos invertidos.
TYPEID	SMALLINT		Identificador de tipo.
SOURCETYPEID	SMALLINT		Identificador de tipo de fuente.
DEFINER	VARCHAR(128)		ID de autorización bajo el cual se ha creado esta correlación de tipos.
LOWER_LEN	INTEGER	Sí	Límite inferior de la longitud/precisión del tipo local.
UPPER_LEN	INTEGER	Sí	Límite superior de la longitud/precisión del tipo local. Si es nulo, el sistema determina el mejor atributo de longitud/precisión.
LOWER_SCALE	SMALLINT	Sí	Límite inferior de la escala para los tipos de datos decimales locales.
UPPER_SCALE	SMALLINT	Sí	Límite superior de la escala para tipos de datos decimales locales. Si es nulo, el sistema determina el mejor atributo de escala.
S_OPR_P	CHAR(2)	Sí	Relación entre la escala local y la precisión local. Se pueden utilizar operadores de comparación básicos. Un nulo indica que no es necesaria ninguna relación específica.
BIT_DATA	CHAR(1)	Sí	Y = El tipo es para datos de bits. N = El tipo no es para datos de bits. NULL = No se trata de un tipo de datos de caracteres o el sistema determina el atributo de datos de bits.
WRAPNAME	VARCHAR(128)	Sí	La correlación se aplica a este protocolo de acceso de datos.
SERVERNAME	VARCHAR(128)	Sí	Nombre de la fuente de datos.
SERVERTYPE	VARCHAR (30)	Sí	La correlación se aplica a este tipo de fuente de datos.
SERVERVERSION	VARCHAR(18)	Sí	La correlación se aplica a esta versión de SERVERTYPE.
REMOTE_TYPESCHEMA	VARCHAR(128)	Sí	Nombre de esquema del tipo remoto.
REMOTE_TYPENAME	VARCHAR(128)		Nombre del tipo de datos tal como está definido en la fuente o fuentes de datos.

Tabla 92. Vista de catálogo SYSCAT.REVTYPEMAPPINGS (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
REMOTE_META_TYPE	CHAR(1)	Sí	S = El tipo remoto es un tipo interno del sistema. T = El tipo remoto es un tipo diferenciado.
REMOTE_LENGTH	INTEGER	Sí	El número máximo de dígitos para el tipo decimal remoto y el número máximo de caracteres para el tipo de caracteres remoto. De lo contrario es nulo.
REMOTE_SCALE	SMALLINT	Sí	El número máximo de dígitos permitidos a la derecha de la coma decimal (para tipos decimales remotos). De lo contrario es nulo.
REMOTE_BIT_DATA	CHAR(1)	Sí	Y = El tipo es para datos de bits. N = El tipo no es para datos de bits. NULL = No se trata de un tipo de datos de caracteres o el sistema determina el atributo de datos de bits.
USER_DEFINED	CHAR(1)		Definido por el usuario.
CREATE_TIME	TIMESTAMP		Hora en que se ha creado la correlación.
REMARKS	VARCHAR(254)	Sí	Comentarios suministrados por el usuario o nulos.

SYSCAT.ROUTINEAUTH

Contiene una o varias filas para cada usuario o grupo al que se ha otorgado el privilegio EXECUTE sobre una rutina determinada de la base de datos.

Tabla 93. Vista de catálogo SYSCAT.ROUTINEAUTH

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
GRANTOR	VARCHAR(128)		ID de autorización del usuario que ha otorgado el privilegio o SYSIBM.
GRANTEE	VARCHAR(128)		ID de autorización del usuario o del grupo que tiene el privilegio.
GRANTEETYPE	CHAR(1)		U = Se otorga a un usuario individual. G = Se otorga a un grupo.
SCHEMA	VARCHAR(128)		Calificador de la rutina.
SPECIFICNAME	VARCHAR(128)	Sí	Nombre específico de la rutina. Si SPECIFICNAME es un valor nulo y ROUTINETYPE no es M, el privilegio se aplica a todas las rutinas del esquema del tipo especificado en ROUTINETYPE. Si SPECIFICNAME es un valor nulo y ROUTINETYPE es M, el privilegio se aplica a todos los métodos del esquema del tipo de sujeto TYPENAME.
TYPESCHEMA	VARCHAR(128)	Sí	Calificador del nombre de tipo del método. Si ROUTINETYPE no es M, TYPESCHEMA es nulo.
TYPENAME	VARCHAR(18)	Sí	Nombre de tipo del método. Si ROUTINETYPE no es M, TYPENAME es nulo. Si TYPENAME es un valor nulo y ROUTINETYPE es M, el privilegio se aplica a los tipos de sujeto del esquema TYPESCHEMA.
ROUTINETYPE	CHAR(1)		Tipo de rutina: F = Función M = Método P = Procedimiento
EXECUTEAUTH	CHAR(1)		Indica si el destinatario de la operación de otorgar posee el privilegio EXECUTE sobre la función o método: Y = Tiene el privilegio. G = Tiene el privilegio y es otorgable. N = No tiene el privilegio.
GRANT_TIME	TIMESTAMP		Hora en que se ha otorgado el privilegio EXECUTE.

SYSCAT.ROUTINEDEP

Cada fila representa una dependencia de una rutina sobre algún otro objeto. (Esta vista de catálogo reemplaza a SYSCAT.FUNCDEP. La otra vista existe pero permanecen igual que en DB2 Versión 7.1.)

Tabla 94. Vista de catálogo SYSCAT.FUNCDEP

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
ROUTINESCHEMA	VARCHAR(128)		Nombre calificado de la rutina.
ROUTINENAME	VARCHAR(128)		
BTYPE	CHAR(1)		Tipo de objeto del que depende la rutina. A = Seudónimo F = Instancia de rutina O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o vistas R = Tipo estructurado S = Tabla de consulta materializada T = Tabla U = Tabla con tipo V = Vista W = Vista con tipo X = Extensión de índice
BSCHEMA	VARCHAR(128)		Nombre calificado del objeto del que depende la función o método (si BTYPE = F, se trata del nombre específico de una rutina).
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Sí	Si BTYPE = O, S, T, U, V ó W, codifica los privilegios de la tabla o la vista que son necesarios para la rutina dependiente. De lo contrario es nulo.

SYSCAT.ROUTINEPARMS

Contiene una fila para cada parámetro o resultado de una rutina definido en SYSCAT.ROUTINES. (Esta vista de catálogo reemplaza a SYSCAT.FUNCPARMS y SYSCAT.PROCPARMS. Las otras vistas existen pero permanecen igual que en DB2 Versión 7.1.)

Tabla 95. Vista de catálogo SYSCAT.ROUTINEPARMS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
ROUTINESCHEMA	VARCHAR(128)		Nombre de rutina calificado.
ROUTINENAME	VARCHAR(128)		
SPECIFICNAME	VARCHAR(128)		El nombre de la instancia de rutina (puede ser generado por el sistema).
PARMNAME	VARCHAR(128)	Sí	Nombre del parámetro o de la columna del resultado, o nulo si no existe ningún nombre.
ROWTYPE	CHAR(1)		B = Parámetro tanto de entrada como de salida C = Resultado después de la conversión del tipo de datos O = Parámetro de salida P = Parámetro de entrada R = Resultado antes de la conversión del tipo de datos
ORDINAL	SMALLINT		Si ROWTYPE = B, O ó P, la posición numérica del parámetro dentro de la signatura de la rutina. Si ROWTYPE = R y la rutina es una función de tabla, la posición numérica de la columna dentro de la tabla resultante. En otro caso, su valor es 0.
TYPESCHEMA	VARCHAR(128)		Nombre calificado de tipo de datos de parámetro o resultado.
TYPENAME	VARCHAR(128)		
LOCATOR	CHAR(1)		Y = El parámetro o resultado se pasa en forma de localizador N = El parámetro o resultado no se pasa en forma de localizador
LENGTH	INTEGER		Longitud del parámetro o resultado. 0 si el parámetro o resultado es un tipo diferenciado. Consulte la Nota 1.
SCALE	SMALLINT		Escala del parámetro o resultado. 0 si el parámetro o resultado es un tipo diferenciado. Consulte la Nota 1.
CODEPAGE	SMALLINT		Página de códigos del parámetro o resultado. 0 indica que no es aplicable o se trata de un parámetro o resultado para datos de caracteres declarados con el atributo FOR BIT DATA.
CAST_FUNCSCHEMA	VARCHAR(128)	Sí	Nombre calificado de la función utilizada para convertir un argumento o resultado. Se aplica a funciones derivadas y externas; de lo contrario nulo.
CAST_FUNCSPECIFIC	VARCHAR(128)	Sí	

Tabla 95. Vista de catálogo SYSCAT.ROUTINEPARMS (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TARGET_TYPESCHEMA	VARCHAR(128)	Sí	Nombre calificado del tipo de destino si el tipo del parámetro o resultado es REFERENCE. Valor nulo si el tipo del parámetro o resultado no es REFERENCE.
TARGET_TYPENAME	VARCHAR(128)	Sí	
SCOPE_TABSCHEMA	VARCHAR(128)	Sí	Nombre calificado del ámbito (tipo de destino) si el tipo del parámetro o resultado es REFERENCE. Valor nulo si el tipo del parámetro o resultado no es REFERENCE o el ámbito no está definido.
SCOPE_TABNAME	VARCHAR(128)	Sí	
TRANSFORMGRPNAME	VARCHAR(128)	Sí	Nombre del grupo transformación para un parámetro o resultado de tipo estructurado.
REMARKS	VARCHAR(254)	Sí	Observaciones de los parámetros.

Notas:

1. LENGTH y SCALE se establecen en 0 para las funciones derivadas (funciones definidas con una referencia a otra función), porque heredan la longitud y escala de los parámetros de su fuente.

SYSCAT.ROUTINES

Contiene una fila para cada función definida por el usuario (escalar, de tabla o de fuente), método generado por el sistema, método definido por el usuario o procedimiento. No incluye las funciones incorporadas. (Esta vista de catálogo reemplaza a SYSCAT.FUNCTIONS y SYSCAT.PROCEDURES. Las otras vistas existen pero permanecen igual que en DB2 Versión 7.1.)

Tabla 96. Vista de catálogo SYSCAT.ROUTINES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
ROUTINESCHEMA	VARCHAR(128)		Nombre de rutina calificado.
ROUTINENAME	VARCHAR(128)		
ROUTINETYPE	CHAR(1)		F = Función M = Método P = Procedimiento.
DEFINER	VARCHAR(128)		ID de autorización del encargado de definir la rutina.
SPECIFICNAME	VARCHAR(128)		El nombre de la instancia de rutina (puede ser generado por el sistema).
ROUTINEID	INTEGER		ID de rutina asignado internamente.
RETURN_TYPESCHEMA	VARCHAR(128)	Sí	Nombre calificado del tipo devuelto por una función escalar o método.
RETURN_TYPENAME	VARCHAR(128)	Sí	
ORIGIN	CHAR(1)		B = interno E = Definido por el usuario, externo M = Plantilla Q = Con cuerpo de SQL U = Definido por el usuario, basado en una fuente S = Generado por el sistema T = Transformación generada por el sistema
FUNCTIONTYPE	CHAR(1)		C = Función de columna R = Función de fila S = Función escalar o método T = Función de tabla Blanco = Procedimiento
PARM_COUNT	SMALLINT		El número de parámetros.
LANGUAGE	CHAR(8)		Lenguaje de implantación del cuerpo de la rutina. Los valores posibles son C, COBOL, JAVA, OLE, OLEDB o SQL. En blanco si ORIGIN no es E ni Q.
SOURCESCHEMA	VARCHAR(128)	Sí	Si ORIGIN = U y la rutina es una función definida por el usuario, contiene el nombre calificado de la función fuente. Si ORIGIN = U y la función fuente es incorporada, SOURCESCHEMA es 'SYSIBM' y SOURCESPECIFIC es 'N/D para incorporada'. Nulo si ORIGIN no es U.
SOURCESPECIFIC	VARCHAR(128)	Sí	

Tabla 96. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
DETERMINISTIC	CHAR(1)		Y = Determinístico (los resultados son coherentes) N = No determinístico (los resultados pueden diferir) En blanco si ORIGIN no es E ni Q.
EXTERNAL_ACTION	CHAR(1)		E = La función tiene efectos adicionales externos (el número de invocaciones es importante) N = Ningún efecto adicional En blanco si ORIGIN no es E ni Q.
NULLCALL	CHAR(1)		Y = CALLED ON NULL INPUT N = RETURNS NULL ON NULL INPUT (el resultado de la función es implícitamente nulo si el operando u operandos son nulos. En blanco si ORIGIN no es E ni Q.
CAST_FUNCTION	CHAR(1)		Y = Es una función de conversión de tipos de datos N = No es una función de conversión de tipos de datos
ASSIGN_FUNCTION	CHAR(1)		Y = Función de asignación implícita N = No es una función de asignación
SCRATCHPAD	CHAR(1)		Y = Esta rutina tiene una memoria de trabajo. N = Esta rutina no tiene una memoria de trabajo. En blanco si ORIGIN no es E o ROUTINETYPE es P.
SCRATCHPAD_LENGTH	SMALLINT		*n* = Longitud de la memoria de trabajo en bytes 0 = SCRATCHPAD es N -1 = LANGUAGE es OLEDB
FINALCALL	CHAR(1)		Y = Se realiza una llamada final a esta función al ejecutar el fin de sentencia. N = No se realiza ninguna llamada final. En blanco si ORIGIN no es E.
PARALLEL	CHAR(1)		Y = La función se puede ejecutar en paralelo. N = La función no se puede ejecutar en paralelo. En blanco si ORIGIN no es E.

Tabla 96. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
PARAMETER_STYLE	CHAR(8)		Indica el estilo de parámetro declarado al crear la rutina. Valores: DB2SQL SQL DB2GENRL GENERAL JAVA DB2DARI GNRLNULL En blanco si ORIGIN no es E.
FENCED	CHAR(1)		Y = Limitado N = No limitado En blanco si ORIGIN no es E.
SQL_DATA_ACCESS	CHAR(1)		C = CONTAINS SQL: sólo se permite SQL que no lea ni modifique datos SQL. M = MODIFIES SQL DATA: se permite todo tipo de SQL en las rutinas. N = NO SQL: no se permite SQL. R = READS SQL DATA: sólo se permite SQL que lea datos SQL.
DBINFO	CHAR(1)		Y = DBINFO se pasa. N = DBINFO no se pasa.
PROGRAMTYPE	CHAR(1)		M = Principal S = Subrutina
COMMIT_ON_RETURN	CHAR(1)		N = Los cambios no se confirman una vez finalizado el procedimiento. En blanco si ROUTINETYPE no es P.
RESULT_SETS	SMALLINT		El límite superior estimado de los conjuntos del resultado devueltos.
SPEC_REG	CHAR(1)		I = INHERIT SPECIAL REGISTERS: los registros especiales empiezan por sus valores a partir de la sentencia que los invoca. En blanco si ORIGIN no es E ni Q.
FEDERATED	CHAR(1)		No utilizado.
THREADSAFE	CHAR(1)		Y = La rutina puede ejecutarse en el mismo proceso que las otras rutinas. N = La rutina debe ejecutarse en un proceso independiente del de las otras rutinas. En blanco si ORIGIN no es E.

Tabla 96. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
VALID	CHAR(1)		Y = el procedimiento SQL es válido. N = el procedimiento SQL no es válido. X = el procedimiento SQL no es operativo porque se ha descartado algún objeto que necesita. Se debe descartar y volver a crear explícitamente el procedimiento SQL. En blanco si ORIGIN no es Q.
METHODIMPLEMENTED	CHAR(1)		Y = El método está implementada. N = Especificación de método sin una implementación. En blanco si ROUTINETYPE no es M.
METHODEFFECT	CHAR(2)		MU = Método mutador OB = Método observador CN = Método constructor En blanco si FUNCTIONTYPE no es T.
TYPE_PRESERVING	CHAR(1)		Y = El tipo de retorno está determinado por un parámetro que conserva el tipo. Todos los métodos mutadores generados por el sistema conservan el tipo. N = El tipo de retorno es el tipo de retorno declarado del método. En blanco si ROUTINETYPE no es M.
WITH_FUNC_ACCESS	CHAR(1)		Y = Este método se puede invocar utilizando una notación funcional. N = Este método no se puede invocar utilizando una notación funcional. En blanco si ROUTINETYPE no es M.
OVERRIDEN_METHODID	INTEGER	Sí	Reservado para su utilización en el futuro.
SUBJECT_TYPESCHEMA	VARCHAR(128)	Sí	Tipo sujeto para el método.
SUBJECT_TYPENAME	VARCHAR(128)	Sí	
CLASS	VARCHAR(128)	Sí	Si LANGUAGE = JAVA, identifica la clase que implanta esta rutina. De lo contrario, es nulo.
JAR_ID	VARCHAR(128)	Sí	Si LANGUAGE = JAVA, identifica el archivo jar que implanta esta rutina. De lo contrario, es nulo.
JARSCHEMA	VARCHAR(128)	Sí	Si LANGUAGE = JAVA, identifica el esquema del archivo jar que implanta esta rutina. De lo contrario, es nulo.
JAR_SIGNATURE	VARCHAR(1024)	Sí	Si LANGUAGE = JAVA, identifica la signatura del método Java que implanta esta rutina. De lo contrario, es nulo.
CREATE_TIME	TIMESTAMP		Indicación de la fecha y hora de creación de la rutina. Se establece en 0 para funciones de la Versión 1.

Tabla 96. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
ALTER_TIME	TIMESTAMP		Indicación de la fecha y hora más reciente de alteración de la rutina. Si la rutina no se ha alterado, establézcala en CREATE_TIME.
FUNC_PATH	VARCHAR(254)	Sí	Vía de acceso de SQL en el momento en que se definió la rutina.
QUALIFIER	VARCHAR(128)		Valor del esquema por omisión en el momento de definición de objeto.
IOS_PER_INVOC	DOUBLE		El número estimado de operaciones de E/S por invocación; -1 si no se conoce (0 por omisión).
INSTS_PER_INVOC	DOUBLE		El número estimado de instrucciones por invocación; -1 si no se conoce (450 por omisión).
IOS_PER_ARGBYTE	DOUBLE		El número estimado de operaciones de E/S por byte de argumento de entrada; -1 si no se conoce (0 por omisión).
INSTS_PER_ARGBYTE	DOUBLE		El número estimado de instrucciones por byte de argumento de entrada; -1 si no se conoce (0 por omisión).
PERCENT_ARGBYTES	SMALLINT		Porcentaje medio estimado de bytes de argumento de entrada que leerá la rutina realmente; -1 si no se conoce (100 por omisión).
INITIAL_IOS	DOUBLE		El número estimado de operaciones de E/S realizadas la primera/última vez que se invoca la rutina; -1 si no se conoce (0 por omisión).
INITIAL_INSTS	DOUBLE		El número estimado de instrucciones ejecutadas la primera/última vez que se invoca la rutina; -1 si no se conoce (0 por omisión).
CARDINALITY	BIGINT		La cardinalidad predicha de una función de tabla; -1 si no se conoce o si la rutina no es una función de tabla.
SELECTIVITY	DOUBLE		Utilizado para predicados definidos por el usuario; -1 si no hay ningún predicado definido por el usuario. Consulte la Nota 1.
RESULT_COLS	SMALLINT		Para una función de tabla (ROUTINETYPE = F y TYPE = T) contiene el número de columnas de la tabla resultante. Para otras funciones y métodos (ROUTINETYPE = F ó M), contiene 1. Para procedimientos (ROUTINETYPE = P), contiene 0.
IMPLEMENTATION	VARCHAR(254)	Sí	Si ORIGIN = E, identifica la vía de acceso/módulo/función que implanta esta función. Si ORIGIN = U y la función fuente es incorporada, esta columna contiene el nombre y signatura de la función fuente. De lo contrario, es nulo.
LIB_ID	INTEGER	Sí	Reservado para su utilización en el futuro.

Tabla 96. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TEXT_BODY_OFFSET	INTEGER		Si LANGUAGE = SQL, el desplazamiento respecto al inicio del cuerpo del procedimiento SQL del texto completo de la sentencia CREATE PROCEDURE; -1 si LANGUAGE no es SQL.
TEXT	CLOB(2M)	Sí	Si LANGUAGE = SQL, el texto de la sentencia CREATE FUNCTION, CREATE METHOD, o CREATE PROCEDURE.
NEWSAVEPOINTLEVEL	CHAR(1)		Indica si la rutina inicia un nivel de punto de salvar nuevo cuando se invoca. Y = Se inicia un nivel de punto de salvar nuevo cuando se invoca la rutina. N = No se inicia un nivel de punto de salvar nuevo cuando se invoca la rutina. La rutina utiliza el nivel de punto de salvar existente. En blanco si ORIGIN no es E ni Q.
DEBUG_MODE	CHAR(3)		OFF = La depuración está desactivada para esta rutina. ON = La depuración está activada para esta rutina.
TRACE_LEVEL	CHAR(1)		Reservado para su utilización en el futuro.
DIAGNOSTIC_LEVEL	CHAR(1)		Reservado para su utilización en el futuro.
CHECKOUT_USERID	VARCHAR(128)	Sí	ID de usuario del usuario que ha realizando una comprobación del objeto. Nulo si no se ha realizado ninguna comprobación.
PRECOMPILE_OPTIONS	VARCHAR(1024)	Sí	Opciones de precompilación especificadas para la rutina.
COMPILE_OPTIONS	VARCHAR(1024)	Sí	Opciones de compilación especificadas para la rutina.
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario o nulo.

Notas:

1. Esta columna se establece en -1 durante la migración en el descriptor empaquetado y los catálogos del sistema para todas las rutinas definidas por el usuario. Para un predicado definido por el usuario, la selectividad es -1 en el catálogo del sistema. En este caso, el valor de selectividad utilizado por el optimizador es 0.01.

SYSCAT.SCHEMAAUTH

Contiene una o varias filas para cada usuario o grupo a los que se ha otorgado un privilegio para un esquema determinado de la base de datos. Todos los privilegios para un esquema individual otorgados por un otorgante específico a un receptor del otorgamiento específico aparecen en una sola fila.

Tabla 97. Vista de catálogo SYSCAT.SCHEMAAUTH

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
GRANTOR	VARCHAR(128)		ID de autorización del usuario que ha otorgado los privilegios o SYSIBM.
GRANTEE	VARCHAR(128)		ID de autorización del usuario o grupo que tiene los privilegios.
GRANTEETYPE	CHAR(1)		U = Se otorga a un usuario individual. G = Se otorga a un grupo.
SCHEMANAME	VARCHAR(128)		Nombre del esquema.
ALTERINAUTH	CHAR(1)		Indica si receptor del otorgamiento tiene el privilegio ALTERIN en el esquema: Y = Tiene el privilegio. G = Tiene el privilegio y es otorgable. N = No tiene el privilegio.
CREATEINAUTH	CHAR(1)		Indica si el receptor del otorgamiento tiene el privilegio CREATEIN en el esquema: Y = Tiene el privilegio. G = Tiene el privilegio y es otorgable. N = No tiene el privilegio.
DROPINAUTH	CHAR(1)		Indica si el receptor del otorgamiento tiene el privilegio DROPIN en el esquema: Y = Tiene el privilegio. G = Tiene el privilegio y es otorgable. N = No tiene el privilegio.

SYSCAT.SCHEMATA

Contiene una fila para cada esquema.

Tabla 98. Vista de catálogo SYSCAT.SCHEMATA

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
SCHEMANAME	VARCHAR(128)		Nombre del esquema.
OWNER	VARCHAR(128)		ID de autorización del esquema. El valor para los esquemas creados implícitamente es SYSIBM.
DEFINER	VARCHAR(128)		Usuario que ha creado el esquema.
CREATE_TIME	TIMESTAMP		Indicación de fecha y hora que indica cuándo se creó el objeto.
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario.

SYSCAT.SEQUENCEAUTH

Contiene una fila para cada ID de autorización que puede utilizarse para utilizar o alterar una secuencia.

Tabla 99. Vista de catálogo SYSCAT.SEQUENCEAUTH

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
GRANTOR	VARCHAR(128)		SYSIBM o ID de autorización que ha otorgado el privilegio.
GRANTEE	VARCHAR(128)		ID de autorización que posee el privilegio.
GRANTEETYPE	CHAR(1)		Tipo de ID de autorización que posee el privilegio. U = Se otorga a un usuario individual
SEQSCHEMA	VARCHAR(128)		Nombre calificado de la secuencia.
SEQNAME	VARCHAR(128)		
USAGEAUTH	CHAR(1)		Y = Tiene el privilegio N = No tiene el privilegio G = Tiene el privilegio y es otorgable
ALTERAUTH	CHAR(1)		Y = Tiene el privilegio N = No tiene el privilegio G = Tiene el privilegio y es otorgable

SYSCAT.SEQUENCES

Contiene una fila para cada columna de secuencia o de identidad definida en la base de datos.

Tabla 100. Columnas de la vista de catálogo SYSCAT.SEQUENCES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
SEQSCHEMA	VARCHAR(128)		Nombre calificado de la secuencia (generado por DB2 para una columna de identificación).
SEQNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		Definidor de la secuencia.
OWNER	VARCHAR(128)		Propietario de la secuencia.
SEQID	INTEGER		ID interno de la secuencia.
SEQTYPE	CHAR(1)		Tipo de secuencia S = Secuencia corriente I = Secuencia de identificación
INCREMENT	DECIMAL(31,0)		Valor de incremento.
START	DECIMAL(31,0)		Valor inicial.
MAXVALUE	DECIMAL(31,0)		Valor máximo.
MINVALUE	DECIMAL(31,0)		Valor mínimo.
CYCLE	CHAR(1)		Indica si se producirá el ciclo al alcanzar un límite: Y - se producirá el ciclo N - no se producirá el ciclo
CACHE	INTEGER		El número de valores de secuencia que se deben preasignar en memoria para el acceso más rápido. 0 indica que no se preasignan valores.
ORDER	CHAR(1)		Indica si los números de secuencia deben generarse según el orden de petición: Y - los números de secuencia deben generarse según el orden de petición N - no es necesario que los números de secuencia se generen según el orden de petición
DATATYPEID	INTEGER		Para tipos internos, ID interno del tipo interno. Para tipos diferenciados, ID interno del tipo diferenciado.
SOURCETYPEID	INTEGER		Para un tipo interno, tiene un valor de 0. Para un tipo diferenciado, es el ID interno del tipo interno que es el tipo fuente para el tipo diferenciado.
CREATE_TIME	TIMESTAMP		Hora en que se ha creado la secuencia.
ALTER_TIME	TIMESTAMP		Hora en que se ha ejecutado la última sentencia ALTER SEQUENCE para esta secuencia.
PRECISION	SMALLINT		Precisión del tipo de datos de la secuencia. Los valores son: 5 para SMALLINT, 10 para INTEGER y 19 para BIGINT. Para DECIMAL, es la precisión del tipo de datos DECIMAL especificado.
ORIGIN	CHAR(1)		Origen de la secuencia U - Secuencia generada por el usuario S - Secuencia generada por el sistema
REMARKS	VARCHAR(254)	Sí	Comentarios proporcionados por el usuario o nulo.

SYSCAT.SERVEROPTIONS

Cada fila contiene las opciones de configuración a nivel de servidor.

Tabla 101. Columnas de la vista de catálogo SYSCAT.SERVEROPTIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
WRAPNAME	VARCHAR(128)	Sí	Nombre de reiniciador.
SERVERNAME	VARCHAR(128)	Sí	Nombre del servidor.
SERVERTYPE	VARCHAR (30)	Sí	Tipo de servidor.
SERVERVERSION	VARCHAR(18)	Sí	Versión de servidor.
CREATE_TIME	TIMESTAMP		Hora en que se ha creado la entrada.
OPTION	VARCHAR(128)		Nombre de la opción de servidor.
SETTING	VARCHAR(2048)		Valor de la opción de servidor.
SERVEROPTIONKEY	VARCHAR(18)		Identifica exclusivamente una fila.
REMARKS	VARCHAR(254)	Sí	Comentarios suministrados por el usuario o nulos.

SYSCAT.SERVERS

Cada fila representa una fuente de datos. Las entradas del catálogo no son necesarias para las tablas que se almacenan en la misma instancia que contiene esta tabla del catálogo.

Tabla 102. Columnas de la vista de catálogo SYSCAT.SERVERS

Nombre	Tipo de datos	Posibilidad de nulos	Descripción
WRAPNAME	VARCHAR(128)		Nombre de reiniciador.
SERVERNAME	VARCHAR(128)		Nombre de la fuente de datos conocida por el sistema.
SERVERTYPE	VARCHAR (30)	Sí	Tipo de fuente de datos (siempre en mayúsculas).
SERVERVERSION	VARCHAR(18)	Sí	Versión de la fuente de datos.
REMARKS	VARCHAR(254)	Sí	Comentarios suministrados por el usuario o nulos.

SYSCAT.STATEMENTS

Contiene una o varias filas para cada sentencia de SQL de cada paquete de la base de datos.

Tabla 103. Vista de catálogo SYSCAT.STATEMENTS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
PKGSCHEMA	VARCHAR(128)		Nombre del paquete.
PKGNAME	CHAR(8)		
UNIQUEID	CHAR(8)		Información de fecha y hora interna que indica cuándo se creó el paquete por primera vez. Útil para identificar un paquete concreto cuando existen varios paquetes con el mismo nombre.
PKGVERSION	VARCHAR(64)		Identificador de versión del paquete.
STMTNO	INTEGER		El número de línea de la sentencia de SQL del módulo fuente del programa de aplicación.
SECTNO	SMALLINT		El número de la sección del paquete que contiene la sentencia de SQL.
SEQNO	SMALLINT		Siempre 1.
TEXT	CLOB(64K)		Texto de la sentencia de SQL.

SYSCAT.TABAUTH

Contiene una o varias filas para cada usuario o grupo a los que se otorga un privilegio para una tabla o vista determinada de la base de datos. Todos los privilegios para una tabla o vista individual otorgados por un otorgante específico a un usuario autorizado específico aparecen en una sola fila.

Tabla 104. Vista de catálogo SYSCAT.TABAUTH

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
GRANTOR	VARCHAR(128)		ID de autorización del usuario que ha otorgado los privilegios o SYSIBM.
GRANTEE	VARCHAR(128)		ID de autorización del usuario o grupo que tiene los privilegios.
GRANTEETYPE	CHAR(1)		U = Se otorga a un usuario individual. G = Se otorga a un grupo.
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla o vista.
TABNAME	VARCHAR(128)		
CONTROLAUTH	CHAR(1)		Indica si el receptor del otorgamiento tiene el privilegio CONTROL en la tabla o vista: Y = Tiene el privilegio. N = No tiene el privilegio.
ALTERAUTH	CHAR(1)		Indica si el receptor del otorgamiento tiene el privilegio ALTER en la tabla: Y = Tiene el privilegio. N = No tiene el privilegio. G = Tiene el privilegio y es otorgable.
DELETEAUTH	CHAR(1)		Indica si el receptor del otorgamiento tiene el privilegio DELETE en la tabla o vista: Y = Tiene el privilegio. N = No tiene el privilegio. G = Tiene el privilegio y es otorgable.
INDEXAUTH	CHAR(1)		Indica si el receptor del otorgamiento tiene el privilegio INDEX en la tabla: Y = Tiene el privilegio. N = No tiene el privilegio. G = Tiene el privilegio y es otorgable.
INSERTAUTH	CHAR(1)		Indica si el receptor del otorgamiento tiene el privilegio INSERT en la tabla o vista: Y = Tiene el privilegio. N = No tiene el privilegio. G = Tiene el privilegio y es otorgable.
SELECTAUTH	CHAR(1)		Indica si el receptor del otorgamiento tiene el privilegio SELECT en la tabla o vista: Y = Tiene el privilegio. N = No tiene el privilegio. G = Tiene el privilegio y es otorgable.

Tabla 104. Vista de catálogo SYSCAT.TABAUTH (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
REFAUTH	CHAR(1)		Indica si el receptor del otorgamiento tiene el privilegio REFERENCE en la tabla o vista: Y = Tiene el privilegio. N = No tiene el privilegio. G = Tiene el privilegio y es otorgable.
UPDATEAUTH	CHAR(1)		Indica si el receptor del otorgamiento tiene el privilegio UPDATE en la tabla o vista: Y = Tiene el privilegio. N = No tiene el privilegio. G = Tiene el privilegio y es otorgable.

SYSCAT.TABCONST

Cada fila representa una restricción de tabla del tipo CHECK, UNIQUE, PRIMARY KEY o FOREIGN KEY.

Tabla 105. Vista de catálogo SYSCAT.TABCONST

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
CONSTNAME	VARCHAR(18)		Nombre de la restricción (exclusivo en una tabla).
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla a la que se aplica esta restricción.
TABNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		ID de autorización bajo el cual se ha definido la restricción.
TYPE	CHAR(1)		Indica el tipo de restricción: F = Clave foránea I = Dependencia funcional K = Comprobación P = Clave primaria U = Unicidad
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario o nulo.
ENFORCED	CHAR(1)		Y = Imponer la restricción N = No imponer la restricción
CHECKEXISTINGDATA	CHAR(1)		D = Diferir la comprobación de los datos existentes I = Comprobación inmediata de los datos existentes N = No comprobar los datos existentes nunca
ENABLEQUERYOPT	CHAR(1)		Y = Optimización de consulta activada N = Optimización de consulta desactivada

SYSCAT.TABDEP

Contiene una fila para cada vez que una vista o una tabla de consulta materializada depende de otro objeto. También codifica la forma en que los privilegios de esta vista dependen de los privilegios de las tablas y vistas principales. (La vista de catálogo VIEWDEP todavía está disponible, pero sólo a nivel de la Versión 7.1).

Tabla 106. Vista de catálogo SYSCAT.TABDEP

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TABSCHEMA	VARCHAR(128)		Nombre de la vista o de la tabla de consulta materializada que depende de una tabla base.
TABNAME	VARCHAR(128)		
DTYPE	CHAR(1)		S = Tabla de consulta materializada V = Vista (de no tipo) W = Vista con tipo
DEFINER	VARCHAR(128)	Sí	ID de autorización del creador de la vista.
BTYPE	CHAR(1)		Tipo de objeto BNAME: A = Seudónimo F = Instancia de función N = Apodo O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas I = Índice si se registra una dependencia de una tabla base R = Tipo estructurado S = Tabla de consulta materializada T = Tabla U = Tabla con tipo V = Vista W = Vista con tipo
BSCHEMA	VARCHAR(128)		Nombre calificado del objeto del que depende la vista.
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Sí	Si BTYPE = N, O, S, T, U, V ó W, codifica los privilegios en la tabla o vista subyacente de la que depende esta vista. De lo contrario es nulo.

SYSCAT.TABLES

Contiene una fila para cada tabla, vista, apodo o seudónimo que se crea. Todas las tablas y vistas de catálogo tienen entradas en la vista de catálogo SYSCAT.TABLES.

Tabla 107. Vista de catálogo SYSCAT.TABLES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla, vista, apodo o seudónimo.
TABNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		Usuario que ha creado la tabla, vista, apodo o seudónimo.
TYPE	CHAR(1)		El tipo de objeto: A = Seudónimo H = Tabla de jerarquía N = Apodo S = Tabla de consulta materializada T = Tabla U = Tabla con tipo V = Vista W = Vista con tipo
STATUS	CHAR(1)		El estado pendiente de comprobación del objeto: N = Tabla, vista, seudónimo o apodo normal C = Pendiente de comprobación en tabla o apodo X = Apodo o vista no operativa
DROPRULE	CHAR(1)		N = Ninguna regla R = La regla restrictiva se aplica al descartar
BASE_TABSCHEMA	VARCHAR(128)	Sí	Si TYPE=A, estas columnas identifican la tabla, vista, seudónimo o apodo a los que hace referencia este seudónimo; de lo contrario son nulos.
BASE_TABNAME	VARCHAR(128)	Sí	
ROWTYPESCHEMA	VARCHAR(128)	Sí	Contiene el nombre calificado del tipo de fila de esta tabla, donde sea aplicable. De lo contrario, es nulo.
ROWTYPENAME	VARCHAR(18)		
CREATE_TIME	TIMESTAMP		La indicación de fecha y hora que indica cuándo se creó el objeto.
STATS_TIME	TIMESTAMP	Sí	La última vez que se ha realizado un cambio en las estadísticas registradas para esta tabla. Nulo si no hay estadísticas disponibles.
COLCOUNT	SMALLINT		El número de columnas en la tabla.
TABLEID	SMALLINT		Identificador interno de tabla.
TBSPACEID	SMALLINT		Identificador interno de espacio de tablas principal para esta tabla.

Tabla 107. Vista de catálogo SYSCAT.TABLES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
CARD	BIGINT		El número total de filas en la tabla. Para tablas en una tabla de jerarquía, su número de filas del nivel determinado de la jerarquía; –1 si no se reúnen estadísticas o la fila describe una vista o seudónimo; –2 para tablas de jerarquía (tablas-H).
NPAGES	INTEGER		El número total de páginas en las que existen las filas de la tabla; –1 si no se reúnen estadísticas o la fila describe una vista o un seudónimo; –2 para subtablas o tablas-H.
FPAGES	INTEGER		El número total de páginas; –1 si no se reúnen estadísticas o la fila describe una vista o un seudónimo; –2 para subtablas o tablas-H.
OVERFLOW	INTEGER		El número total de registros de desbordamiento en la tabla; –1 si no se reúnen estadísticas o la fila describe una vista o un seudónimo; –2 para subtablas o tablas-H.
TBSPACE	VARCHAR(18)	Sí	Nombre de espacio de tablas principal para la tabla. Si no se especifica ningún otro espacio de tablas, todas las partes de la tabla se almacenan en este espacio de tablas. Es nulo para seudónimos y vistas.
INDEX_TBSPACE	VARCHAR(18)	Sí	Nombre del espacio de tablas que contiene todos los índices creados en esta tabla. Nulo para los seudónimos y vistas o si se ha omitido la cláusula INDEX IN o se ha especificado con el mismo valor que la cláusula IN de la sentencia CREATE TABLE.
LONG_TBSPACE	VARCHAR(18)	Sí	Nombre del espacio de tablas que contiene todos los datos largos (tipos de columna LONG o LOB) para esta tabla. Nulo para los seudónimos y vistas, o si se ha omitido la cláusula LONG IN o se ha especificado con el mismo valor que la cláusula IN de la sentencia CREATE TABLE.
PARENTS	SMALLINT	Sí	El número de tablas padre de esta tabla (el número de restricciones de referencia de las que depende esta tabla).
CHILDREN	SMALLINT	Sí	El número de tablas dependientes de esta tabla (el número de restricciones de referencia en las que esta tabla es padre).
SELFREFS	SMALLINT	Sí	El número de restricciones de referencia propia para esta tabla (el número de restricciones de referencia en las que esta tabla es tanto padre como dependiente).
KEYCOLUMNS	SMALLINT	Sí	El número de columnas de la clave primaria de la tabla.

Tabla 107. Vista de catálogo SYSCAT.TABLES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
KEYINDEXID	SMALLINT	Sí	ID de índice del índice principal. Este campo es nulo o 0 si no hay clave primaria.
KEYUNIQUE	SMALLINT		El número de restricciones de unicidad (distintas de la clave primaria) definidas en esta tabla.
CHECKCOUNT	SMALLINT		El número de restricciones de comprobación definidas en esta tabla.
DATACAPTURE	CHAR(1)		Y = La tabla participa en la duplicación de datos N = No participa L = La tabla interviene en la duplicación de datos, incluida la duplicación de las columnas LONG VARCHAR y LONG VARGRAPHIC
CONST_CHECKED	CHAR(32)		El byte 1 representa las restricciones de clave foránea. El byte 2 representa las restricciones de comprobación. El byte 5 representa la tabla de consulta materializada. El byte 6 representa columnas generadas. El byte 7 representa la tabla dispuesta con antelación. Los demás bytes están reservados. Codifica la información de restricción en la comprobación. Valores: Y = Comprobado por el sistema U = Comprobado por el usuario N = No comprobado (pendiente) W = Estaba en un estado 'U' cuando la tabla se puso en estado pendiente de comprobación (pendiente) F = En el byte 5, la tabla de consulta materializada no puede renovarse de forma incremental. En el byte 7, el contenido de la tabla dispuesta con antelación es incompleto y no puede utilizarse para renovaciones incrementales de la tabla de consulta materializada asociada.
PMAP_ID	SMALLINT	Sí	Identificador del mapa de particionamiento utilizado por esta tabla. Es nulo para seudónimos y vistas.

Tabla 107. Vista de catálogo SYSCAT.TABLES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
PARTITION_MODE	CHAR(1)		Modalidad utilizada para tablas de una base particionada. H = Información inservible en la clave de particionamiento R = Tabla duplicada en las particiones de base de datos En blanco para seudónimos, vistas y tablas en grupos de particiones de base de datos de partición única sin clave de particionamiento definida. También en blanco para apodos.
LOG_ATTRIBUTE	CHAR(1)		0 = Registro cronológico por omisión 1 = La tabla que se ha creado inicialmente no se ha anotado cronológicamente
PCTFREE	SMALLINT		Porcentaje de cada página que se ha de reservar para inserciones posteriores. Se puede cambiar por ALTER TABLE.
APPEND_MODE	CHAR(1)		Controla cómo se insertan las filas en las páginas: N = Se insertan filas nuevas en espacios existentes si están disponibles Y = Se añaden filas nuevas al final de los datos Valor inicial es N.
REFRESH	CHAR(1)		Modalidad de renovación: D = Diferida I = Inmediata O = Una vez En blanco si no es una tabla de consulta materializada
REFRESH_TIME	TIMESTAMP	Sí	Para REFRESH = D ó O, indicación de fecha y hora de la sentencia REFRESH TABLE que renovó los datos por última vez. De lo contrario es nulo.
LOCKSIZE	CHAR(1)		Indica la granularidad de bloqueo preferente para tablas cuando son accedidas mediante sentencias DML. Sólo se aplica a tablas. Los valores posibles son: R = Fila T = Tabla En blanco si no es aplicable El valor inicial es R.
VOLATILE	CHAR(1)		C = La cardinalidad de la tabla es volátil En blanco si no es aplicable
ROW_FORMAT	CHAR(1)		No utilizado.

Tabla 107. Vista de catálogo SYSCAT.TABLES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
PROPERTY	VARCHAR(32)		Propiedades de la tabla. Un solo blanco indica que la tabla no tiene propiedades.
STATISTICS_PROFILE	CLOB(32K)	Sí	Mandato RUNSTATS utilizado para registrar un perfil estadístico de la tabla.
COMPRESSION	CHAR(1)		V = La compresión de los valores está activada y se utiliza un formato de fila que da soporte a la compresión N = Sin compresión. Se utiliza un formato de fila que no da soporte a la compresión
ACCESS_MODE	CHAR(1)		Modalidad de acceso del objeto. Esta modalidad de acceso se utiliza junto con el campo STATUS para representar un estado de los cuatro posibles. Los valores posibles son: N = Sin acceso (se corresponde con un valor de estado de C) R = Sólo lectura (se corresponde con un valor de estado de C) D = Sin movimiento de datos (se corresponde con un valor de estado de N) F = Acceso completo (se corresponde con un valor de estado de N)
CLUSTERED	CHAR(1)	Sí	Y = tabla con clústeres de varias dimensiones (MDC) Nulo para un tabla que no sea MDC
ACTIVE_BLOCKS	INTEGER	Sí	El número total de páginas de bloques en uso en una tabla MDC; –1 si no se reúnen estadísticas.
MAXFREESPACESEARCH	SMALLINT		Reservado para su utilización en el futuro.
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario.

SYSCAT.TABLESPACES

Contiene una fila para cada espacio de tablas.

Tabla 108. Vista de catálogo SYSCAT.TABLESPACES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TBSPACE	VARCHAR(18)		Nombre del espacio de tablas.
DEFINER	VARCHAR(128)		ID de autorización del encargado de definir el espacio de tablas.
CREATE_TIME	TIMESTAMP		Hora de creación del espacio de tablas.
TBSPACEID	INTEGER		Identificador interno del espacio de tablas.
TBSPACETYPE	CHAR(1)		El tipo del espacio de tablas: S = Espacio gestionado por el sistema D = Espacio gestionado por la base de datos
DATATYPE	CHAR(1)		El tipo de datos que se puede almacenar: A = Todos los tipos de datos permanentes L = Datos grandes - datos largos o datos de índices T = Sólo tablas temporales del sistema U = Sólo tablas temporales declaradas
EXTENTSIZE	INTEGER		El tamaño de la extensión, expresado en páginas de tamaño PAGESIZE. Este volumen de páginas se escribe en un contenedor individual del espacio de tablas antes de cambiar al contenedor siguiente.
PREFETCHSIZE	INTEGER		El número de páginas de tamaño PAGESIZE que se pueden leer cuando se efectúa una captación previa. -1 si el tamaño de la captación previa es AUTOMATIC.
OVERHEAD	DOUBLE		Actividad general del controlador y tiempo de latencia y de búsqueda en disco, en milisegundos.
TRANSFERRATE	DOUBLE		Tiempo para leer una página de tamaño PAGESIZE en el almacenamiento intermedio.
PAGESIZE	INTEGER		Tamaño (en bytes) de páginas en el espacio de tablas.
DBPGNAME	VARCHAR(18)		Nombre del grupo de particiones de base de datos para el espacio de tablas.
BUFFERPOOLID	INTEGER		ID de la agrupación de almacenamientos intermedios utilizada por este espacio de tablas; 1 indica la agrupación de almacenamientos intermedios por omisión.
DROP_RECOVERY	CHAR(1)		N = la tabla no es recuperable después de una sentencia DROP TABLE Y = la tabla es recuperable después de una sentencia DROP TABLE
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario.

SYSCAT.TABOPTIONS

Cada fila contiene la opción asociada con una tabla remota.

Tabla 109. Vista de catálogo SYSCAT.TABOPTIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TABSCHEMA	VARCHAR(128)		Nombre calificado de tabla, vista, seudónimo o apodo.
TABNAME	VARCHAR(128)		
OPTION	VARCHAR(128)		Nombre de la opción de tabla, vista, seudónimo o apodo.
SETTING	VARCHAR(255)		Valor.

SYSCAT.TBSPACEAUTH

Contiene una fila para cada usuario o grupo al que se ha otorgado privilegio USE para un espacio de tablas determinado de la base de datos.

Tabla 110. Vista de catálogo SYSCAT.TBSPACEAUTH

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
GRANTOR	CHAR(128)		ID de autorización del usuario que ha otorgado los privilegios o SYSIBM.
GRANTEE	CHAR(128)		ID de autorización del usuario o grupo que tiene los privilegios.
GRANTEETYPE	CHAR(1)		U = Se otorga a un usuario individual. G = Se otorga a un grupo.
TBSPACE	VARCHAR(18)		Nombre del espacio de tablas.
USEAUTH	CHAR(1)		Indica si el receptor del otorgamiento tiene el privilegio USE en el espacio de tabla: G = Tiene el privilegio y es otorgable. N = No tiene el privilegio. Y = Tiene el privilegio.

SYSCAT.TRANSFORMS

Contiene una fila para cada tipo de función de transformación dentro de un tipo definido por el usuario contenido en un grupo de transformación especificado.

Tabla 111. Vista de catálogo SYSCAT.TRANSFORMS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TYPEID	SMALLINT		ID interno de tipo, tal como está definido en SYSCAT.DATATYPES
TYPESCHEMA	VARCHAR(128)		Nombre calificado del tipo estructurado proporcionado, definido por el usuario.
TYPENAME	VARCHAR(128)		
GROUPNAME	VARCHAR(128)		Nombre del grupo de transformación.
FUNCID	INTEGER	Sí	ID interno de rutina para la función de transformación asociada, tal como está definido en SYSCAT.ROUTINES. Nulo sólo para funciones incorporadas del sistema.
FUNCSCHEMA	VARCHAR(128)		Nombre calificado de las funciones de transformación asociadas.
FUNCNAME	VARCHAR(128)		
SPECIFICNAME	VARCHAR(128)		Nombre específico de la función (instancia).
TRANSFORMTYPE	VARCHAR(8)		'FROM SQL' = La función de transformación transforma un tipo estructurado desde SQL 'TO SQL' = La función de transformación transforma un tipo estructurado a SQL
FORMAT	CHAR(1)		'U' = Definido por el usuario
MAXLENGTH	INTEGER	Sí	Longitud máxima (en bytes) de la salida de la transformación FROM SQL. Nulo para las transformaciones TO SQL.
ORIGIN	CHAR(1)		'O' = Grupo de transformación original (definido por el usuario o el sistema) 'R' = Redefinido
REMARKS	VARCHAR(254)	Sí	Comentarios suministrados por el usuario o nulos.

SYSCAT.TRIGDEP

Contiene una fila para cada dependencia que un activador tiene de otro objeto.

Tabla 112. Vista de catálogo SYSCAT.TRIGDEP

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TRIGSCHEMA	VARCHAR(128)		Nombre calificado del activador.
TRIGNAME	VARCHAR(18)		
BTYPE	CHAR(1)		Tipo de objeto BNAME: A = Seudónimo B = Activador F = Instancia de función N = Apodo O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas R = Tipo estructurado S = Tabla de consulta materializada T = Tabla U = Tabla con tipo V = Vista W = Vista con tipo X = Extensión de índice
BSCHEMA	VARCHAR(128)		Nombre calificado del objeto del que depende un activador.
BNAME	VARCHAR(128)		
TABAUTH	SMALLINT	Sí	Si BTYPE= O, S, T, U, V o W codifica los privilegios en la tabla o vista que necesita este activador; de lo contrario, es nulo.

SYSCAT.TRIGGERS

Contiene una fila para cada activador. Para jerarquías de tablas, cada activador se registra sólo al nivel de la jerarquía donde se ha creado.

Tabla 113. Vista de catálogo SYSCAT.TRIGGERS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TRIGSCHEMA	VARCHAR(128)		Nombre calificado del activador.
TRIGNAME	VARCHAR(18)		
DEFINER	VARCHAR(128)		ID de autorización bajo el cual se ha definido el activador.
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla o vista a la que se aplica este activador.
TABNAME	VARCHAR(128)		
TRIGTIME	CHAR(1)		Momento en que se aplican las acciones activadas a la base de datos, en relación al suceso que ha disparado el activador: A = Activador aplicado después del suceso B = Activador aplicado antes del suceso I = Activador aplicado en lugar del suceso
TRIGEVENT	CHAR(1)		Suceso que dispara el activador. I = Inserción D = Supresión U = Actualización
GRANULARITY	CHAR(1)		El activador se ejecuta una vez por: S = Sentencia R = Fila
VALID	CHAR(1)		Y = El activador es válido X = El activador no es operativo; debe volverse a crear.
CREATE_TIME	TIMESTAMP		Hora en la que se ha definido el activador. Utilizada para resolver las funciones y tipos.
QUALIFIER	VARCHAR(128)		Contiene el valor del esquema por omisión en el momento de la definición de objeto.
FUNC_PATH	VARCHAR(254)		Vía de acceso de función en el momento en que se ha definido el activador. Utilizada para resolver las funciones y tipos.
TEXT	CLOB(64K)		El texto completo de la sentencia CREATE TRIGGER, tal como se ha escrito.
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario o nulo.

SYSCAT.TYPEMAPPINGS

Cada fila contiene una correlación definida por el usuario de un tipo de datos interno remoto con un tipo de datos interno local.

Tabla 114. Vista de catálogo SYSCAT.TYPEMAPPINGS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
TYPE_MAPPING	VARCHAR(18)		Nombre de la correlación de tipos (puede ser generado por el sistema).
TYPESCHEMA	VARCHAR(128)	Sí	Nombre de esquema del tipo. Nulo para tipos internos del sistema.
TYPENAME	VARCHAR(18)		Nombre del tipo local en una correlación de tipos de datos.
TYPEID	SMALLINT		Identificador de tipo.
SOURCETYPEID	SMALLINT		Identificador de tipo de fuente.
DEFINER	VARCHAR(128)		ID de autorización bajo el cual se ha creado esta correlación de tipos.
LENGTH	INTEGER	Sí	Longitud o precisión máxima del tipo de datos. Si es nulo, el sistema determina la mejor longitud/precisión.
SCALE	SMALLINT	Sí	Escala para campos DECIMAL. Si es nulo, el sistema determina el mejor atributo de escala.
BIT_DATA	CHAR(1)	Sí	Y = El tipo es para datos de bits. N = El tipo no es para datos de bits. NULL = No se trata de un tipo de datos de caracteres o el sistema determina el atributo de datos de bits.
WRAPNAME	VARCHAR(128)	Sí	La correlación se aplica a este reiniciador.
SERVERNAME	VARCHAR(128)	Sí	Nombre de la fuente de datos.
SERVERTYPE	VARCHAR (30)	Sí	La correlación se aplica a este tipo de fuente de datos.
SERVERVERSION	VARCHAR(18)	Sí	La correlación se aplica a esta versión de fuente de datos con el tipo especificado en SERVERTYPE.
REMOTE_TYPESCHEMA	VARCHAR(128)	Sí	Nombre de esquema del tipo remoto.
REMOTE_TYPENAME	VARCHAR(128)		Nombre del tipo de datos tal como está definido en la fuente o fuentes de datos.
REMOTE_META_TYPE	CHAR(1)	Sí	S = El tipo remoto es un tipo interno del sistema. T = El tipo remoto es un tipo diferenciado.
REMOTE_LOWER_LEN	INTEGER	Sí	Límite inferior de la longitud o la precisión del tipo decimal remoto. Para los tipos de datos de caracteres, este campo indica el número de caracteres. Un valor de -1 indica que se utiliza la longitud o la precisión por omisión o que el tipo remoto carece de longitud o precisión.

Tabla 114. Vista de catálogo SYSCAT.TYPEMAPPINGS (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
REMOTE_UPPER_LEN	INTEGER	Sí	Límite superior de la longitud o la precisión del tipo decimal remoto. Para los tipos de datos de caracteres, este campo indica el número de caracteres. Un valor de -1 indica que se utiliza la longitud o la precisión por omisión o que el tipo remoto carece de longitud o precisión.
REMOTE_LOWER_SCALE	SMALLINT	Sí	Límite inferior de la escala del tipo remoto.
REMOTE_UPPER_SCALE	SMALLINT	Sí	Límite superior de la escala del tipo remoto.
REMOTE_S_OPR_P	CHAR(2)	Sí	Relación entre la escala remota y la precisión remota. Se pueden utilizar operadores de comparación básicos. Un valor nulo indica que no es necesaria ninguna relación específica.
REMOTE_BIT_DATA	CHAR(1)	Sí	Y = El tipo es para datos de bits. N = El tipo no es para datos de bits. NULL = No se trata de un tipo de datos de caracteres o el sistema determinará el atributo de datos de bits.
USER_DEFINED	CHAR(1)		Definición suministrada por el usuario.
CREATE_TIME	TIMESTAMP		Hora en que se ha creado esta correlación.
REMARKS	VARCHAR(254)	Sí	Comentarios suministrados por el usuario o nulos.

SYSCAT.USEROPTIONS

Cada fila contiene los valores de las opciones específicas de servidor.

Tabla 115. Vista de catálogo SYSCAT.USEROPTIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
AUTHID	VARCHAR(128)		ID de autorización local (siempre en mayúsculas)
SERVERNAME	VARCHAR(128)		Nombre del servidor para el cual se define el usuario.
OPTION	VARCHAR(128)		Nombre de las opciones de usuario.
SETTING	VARCHAR(255)		Valor.

SYSCAT.VIEWS

Contiene una o varias filas para cada vista que se crea.

Tabla 116. Vista de catálogo SYSCAT.VIEWS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
VIEWSCHEMA	VARCHAR(128)		Nombre calificado de una vista o el nombre calificado de una tabla que se utiliza para definir una tabla de consulta materializada o una tabla dispuesta con antelación.
VIEWNAME	VARCHAR(128)		
DEFINER	VARCHAR(128)		ID de autorización del creador de la vista.
SEQNO	SMALLINT		Siempre 1.
VIEWCHECK	CHAR(1)		Indica el tipo de comprobación de la vista: N = Sin opción de comprobación L = Opción de comprobación local C = Opción de comprobación en cascada
READONLY	CHAR(1)		Y = La vista es de sólo lectura debido a su definición. N = La vista no es de sólo lectura.
VALID	CHAR(1)		Y = La definición de una tabla de consulta materializada o una vista es válida. X = La definición de una tabla de consulta materializada o una vista no es operativa; debe volverse a crear.
QUALIFIER	VARCHAR(128)		Contiene el valor del esquema por omisión en el momento de la definición de objeto.
FUNC_PATH	VARCHAR(254)		La vía de acceso de SQL del creador de una vista en el momento en que se ha definido la vista. Cuando se utiliza la vista en sentencias de manipulación de datos, debe utilizarse esta vía de acceso para resolver las llamadas a funciones de la vista. SYSIBM para las vistas creadas antes de la Versión 2.
TEXT	CLOB(64k)		Texto de la sentencia CREATE VIEW.

SYSCAT.WRAPOPTIONS

Cada fila contiene las opciones específicas de reiniciador.

Tabla 117. Vista de catálogo SYSCAT.WRAPOPTIONS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
WRAPNAME	VARCHAR(128)		Nombre de reiniciador.
OPTION	VARCHAR(128)		Nombre de opción de reiniciador.
SETTING	VARCHAR(255)		Valor.

SYSCAT.WRAPPERS

Cada fila contiene información sobre el reiniciador registrado.

Tabla 118. Vista de catálogo SYSCAT.WRAPPERS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción
WRAPNAME	VARCHAR(128)		Nombre de reiniciador.
WRAPTYPE	CHAR(1)		N = No relacional R = Relacional
WRAPVERSION	INTEGER		Versión del reiniciador.
LIBRARY	VARCHAR(255)		Nombre del archivo que contiene el código utilizado para comunicarse con las fuentes de datos asociadas a este reiniciador.
REMARKS	VARCHAR(254)	Sí	Comentario suministrado por el usuario o nulo.

SYSSTAT.COLDIST

Cada fila describe el valor n más frecuente o el valor cuantil n de alguna columna. Las estadísticas no se registran para columnas heredadas de tablas con tipo.

Tabla 119. Vista de catálogo SYSSTAT.COLDIST

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción	Actualizable
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla a la que se aplica esta entrada.	
TABNAME	VARCHAR(128)			
COLNAME	VARCHAR(128)		Nombre de la columna a la que se aplica esta entrada.	
TYPE	CHAR(1)		Tipo de estadísticas reunidas: F = Frecuencia (valor más frecuente) Q = Valor cuantil	
SEQNO	SMALLINT		Si TYPE = F, entonces N en esta columna identifica el valor N más frecuente. Si TYPE=Q, entonces N en esta columna identifica el valor N cuantil.	
COLVALUE	VARCHAR(254)	Sí	El valor de datos, como un literal de caracteres o un valor nulo. Consulte la Nota 1. Esta columna se puede actualizar con una representación válida del valor adecuado para la columna a la que las estadísticas están asociadas. Si el valor de frecuencia necesario es nulo, la columna debe establecerse en NULL.	Sí
VALCOUNT	BIGINT		Si TYPE = F, entonces VALCOUNT es el número de ocurrencias de COLVALUE en la columna. Si TYPE = Q, entonces VALCOUNT es el número de filas cuyo valor es menor o igual que COLVALUE. Esta columna sólo puede actualizarse con los valores siguientes: • >= 0 (cero)	Sí
DISTCOUNT	BIGINT		Si TYPE=q, esta columna registra el número de valores diferenciados que son menores o iguales que COLVALUE (nulo si no está disponible). el número de filas cuyo valor es menor o igual que COLVALUE.	Sí

Tabla 119. Vista de catálogo SYSSTAT.COLDIST (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción	Actualizable

Notas:

1. En la vista de catálogo, el valor de COLVALUE siempre se muestra en la página de códigos de la base de datos y puede contener caracteres de sustitución. Sin embargo, las estadísticas se reúnen internamente en la página de códigos de la tabla de la columna y, por tanto, utilizarán los valores reales de la columna cuando se apliquen durante la optimización de la consulta.

   Si el valor se modifica utilizando una sentencia UPDATE, sólo pueden utilizarse los caracteres de la página de códigos de la base de datos. En caso contrario, al aplicar estadísticas durante la optimización de la consulta se crearán y utilizarán caracteres de sustitución.

SYSSTAT.COLUMNS

Contiene una fila para cada columna cuyas estadísticas pueden actualizarse. Las estadísticas no se registran para columnas heredadas de tablas con tipo.

Tabla 120. Vista de catálogo SYSSTAT.COLUMNS

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción	Actualizable
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla que contiene la columna.	
TABNAME	VARCHAR(128)			
COLNAME	VARCHAR(128)		Nombre de columna.	
COLCARD	BIGINT		El número de valores diferenciados en la columna: –1 si no se reúnen estadísticas; –2 para columnas heredadas y columnas de tablas-H. Para cualquier columna, COLCARD no puede tener un valor superior a la cardinalidad de la tabla que contiene dicha columna. Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó >= 0 (cero)	Sí
HIGH2KEY	VARCHAR(33)	Sí	Segundo nivel más alto de la columna. Este campo está vacío si no se reúnen estadísticas, para columnas heredadas y columnas de tablas-H. Esta columna se puede actualizar con una representación válida del valor que sea adecuado para la columna a la que las estadísticas están asociadas. Vea las notas 1 y 2.	Sí
LOW2KEY	VARCHAR(33)	Sí	Segundo nivel más bajo de la columna. Este campo está vacío si no se reúnen estadísticas, para columnas heredadas y columnas de tablas-H. Esta columna se puede actualizar con una representación válida del valor que sea adecuado para la columna a la que las estadísticas están asociadas. Vea las notas 1 y 2.	Sí
AVGCOLLEN	INTEGER		Longitud promedio de columna. –1 si es un campo largo o LOB o bien si no se han reunido estadísticas; –2 para columnas heredadas y columnas de tablas-H. Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó >= 0 (cero)	Sí
NUMNULLS	BIGINT		Contiene el número de nulos en una columna. –1 si no se reúnen estadísticas. Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó >= 0 (cero)	Sí

Tabla 120. Vista de catálogo SYSSTAT.COLUMNS (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción	Actualizable
SUB_COUNT	SMALLINT		Número medio de subelementos. Aplicable a columnas de caracteres solamente. Por ejemplo, considere la serie siguiente:'datos simulación analítica inteligencia empresarial'. En este ejemplo, SUB_COUNT = 5, porque hay cinco subelementos en la serie.	Sí
SUB_DELIM_ LENGTH	SMALLINT		Longitud media de cada delimitador que separa cada subelemento. Aplicable a columnas de caracteres solamente. Por ejemplo, considere la serie siguiente:'datos simulación analítica inteligencia empresarial'. En este ejemplo, SUB_DELIM_LENGTH = 1, porque cada delimitador es un solo blanco.	Sí

Notas:

1. HIGH2KEY y LOW2KEY representan una versión en ASCII del segundo valor más alto y segundo valor más bajo de una columna respectivamente. Una representación en ASCII de un valor entero puede tener una longitud entre 1 y 11 caracteres, incluido el signo. Los valores HIGH2KEY y LOW2KEY se almacenan de forma que puedan utilizarse en la cláusula SET de una sentencia UPDATE. Para las series de caracteres, ello significa que se añaden comillas simples al principio y al final de la serie y se inserta una comilla simple adicional antes de cada comilla simple que ya esté contenida en la serie. Sólo se tendrán en cuenta los 33 primeros caracteres del valor de la columna.

   Si piensa rellenar los valores HIGH2KEY y LOW2KEY manualmente, asegúrese de lo siguiente:

   - HIGH2KEY y LOW2KEY deben ser valores válidos para el tipo de datos de la columna del usuario correspondiente.
   - La longitud de los valores de HIGH2KEY y LOW2KEY debe ser la menor de: la longitud máxima del tipo de datos de la columna destino o 33, sin incluir las comillas simples adicionales, que pueden aumentar la longitud de la serie hasta 68.
   - HIGH2KEY debe ser mayor que LOW2KEY siempre que haya tres o más valores distintos en la columna correspondiente. En el caso de que haya menos de tres valores distintos en la columna, HIGH2KEY puede ser igual a LOW2KEY.

2. En la vista de catálogo, los valores de HIGH2KEY y de LOW2KEY siempre se muestran en la página de códigos de la base de datos y pueden contener caracteres de sustitución. Sin embargo, las estadísticas se reúnen internamente en la página de códigos de la tabla de la columna y, por tanto, utilizarán los valores reales de la columna cuando se apliquen durante la optimización de la consulta.

   Si los valores se modifican utilizando una sentencia UPDATE, sólo pueden utilizarse los caracteres de la página de códigos de la base de datos. En caso contrario, al aplicar estadísticas durante la optimización de la consulta se crearán y utilizarán caracteres de sustitución.

SYSSTAT.INDEXES

Contiene una fila para cada índice que se define para una tabla.

Tabla 121. Vista de catálogo SYSSTAT.INDEXES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción	Actualizable
INDSCHEMA	VARCHAR(128)		Nombre calificado del índice.	
INDNAME	VARCHAR(18)			
TABSCHEMA	VARCHAR(128)		Calificador del nombre de tabla.	
TABNAME	VARCHAR(128)		Nombre de la tabla o apodo en el que se define el índice.	
COLNAMES	CLOB(1M)		Lista de nombres de columna con prefijos + o −.	
NLEAF	INTEGER		El número de páginas de índice; −1 si no se reúnen estadísticas. Esta columna sólo puede actualizarse con los valores siguientes: • −1 ó > 0 (cero)	Sí
NLEVELS	SMALLINT		Número de niveles de índice; −1 si no se reúnen estadísticas. Esta columna sólo puede actualizarse con los valores siguientes: • −1 ó > 0 (cero)	Sí
FIRSTKEYCARD	BIGINT		El número de valores de primera clave diferenciada; −1 si no se reúnen estadísticas. Esta columna sólo puede actualizarse con los valores siguientes: • −1 ó >= 0 (cero)	Sí
FIRST2KEYCARD	BIGINT		El número de claves diferenciadas que utilizan las dos primeras columnas del índice (−1 si no hay datos estadísticos o no son aplicables). Esta columna sólo puede actualizarse con los valores siguientes: • −1 ó >= 0 (cero)	Sí
FIRST3KEYCARD	BIGINT		El número de claves diferenciadas que utilizan las tres primeras columnas del índice (−1 si no hay datos estadísticos o no son aplicables). Esta columna sólo puede actualizarse con los valores siguientes: • −1 ó >= 0 (cero)	Sí

Tabla 121. Vista de catálogo SYSSTAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción	Actualizable
FIRST4KEYCARD	BIGINT		El número de claves diferenciadas que utilizan las cuatro primeras columnas del índice (–1 si no hay datos estadísticos o no son aplicables). Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó >= 0 (cero)	Sí
FULLKEYCARD	BIGINT		El número de valores de clave completa diferenciada; –1 si no se reúnen estadísticas. Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó >= 0 (cero)	Sí
CLUSTERRATIO	SMALLINT		El optimizador utiliza esta columna. Indica el nivel de clúster de los datos del índice; –1 si no se reúnen estadísticas o si se han reunido estadísticas detalladas de índice. Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó entre 0 y 100	Sí
CLUSTERFACTOR	DOUBLE		El optimizador utiliza esta columna. Es una mejor medición del nivel de clúster ó –1 si no se han reunido estadísticas de índice detalladas. Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó entre 0 y 1	Sí
SEQUENTIAL_PAGES	INTEGER		El número de páginas ubicadas en disco por orden de clave de índice, con pocos o ningún hueco entre ellas; –1 si no hay datos estadísticos disponibles. Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó >= 0 (cero)	Sí
DENSITY	INTEGER		Proporción de SEQUENTIAL_PAGES para numerar las páginas del rango de páginas ocupadas por el índice, expresada como un porcentaje (entero entre 0 y 100; –1 si no hay datos estadísticos disponibles.) Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó entre 0 y 100	Sí

Tabla 121. Vista de catálogo SYSSTAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción	Actualizable
PAGE_FETCH_PAIRS	VARCHAR(254)		Una lista de pares de enteros, representada en la forma de caracteres. Cada par representa el número de páginas de un almacenamiento intermedio hipotético, el número de lecturas de páginas necesario para explorar el índice utilizando dicho almacenamiento hipotético. (Serie de longitud cero si no hay datos disponibles.) Esta columna puede actualizarse con los valores de entrada siguientes: • El delimitador de pares y los caracteres separadores de pares son los únicos caracteres no numéricos aceptados. • Los blancos son los únicos caracteres reconocidos como delimitadores de pares y separadores de pares. • Cada entrada numérica debe tener una entrada numérica asociada que le acompañe y ambas deben ir separadas por el carácter separador de pares. • Cada par debe estar separado de los otros pares por el carácter delimitador de pares. • Cada entrada numérica esperada debe estar entre 0 y 9 (sólo valores positivos).	Sí
NUMRIDS	BIGINT		El número de RID del índice; –1 si no se reúnen estadísticas.	Sí
NUMRIDS_DELETED	BIGINT		El número de RID del índice que están marcados para su supresión, excluidos aquellos de páginas en las que todos los RID están marcados para su supresión; –1 si no se reúnen estadísticas.	Sí
NUM_EMPTY_LEAFS	BIGINT		El número de páginas del índice en las que todos los RID están marcados para su supresión; –1 si no se reúnen estadísticas.	Sí
AVERAGE_RANDOM_FETCH_PAGES	DOUBLE		Promedio de páginas de la tabla aleatorias entre los accesos a páginas secuenciales al captar utilizando el índice; –1 si no se conoce. Consulte las notas 1 y 2.	
AVERAGE_RANDOM_PAGES	DOUBLE		Promedio de páginas del índice aleatorias entre los accesos a páginas del índice; –1 si no se conoce. Consulte la nota 2.	
AVERAGE_SEQUENCE_GAP	DOUBLE		Espacio entre las secuencias de páginas del índice. Detectado mediante una exploración de las páginas del índice, cada espacio representa el promedio de páginas del índice que deben captarse de forma aleatoria entre las secuencias de las páginas del índice; –1 si no se conoce. Consulte la nota 2.	

Tabla 121. Vista de catálogo SYSSTAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción	Actualizable
AVERAGE_SEQUENCE_FETCH_GAP	DOUBLE		Espacio entre las secuencias de páginas de la tabla al captar utilizando el índice. Detectado mediante una exploración de las páginas del índice, cada espacio representa el promedio de páginas de la tabla que deben captarse de forma aleatoria entre las secuencias de las páginas de la tabla; –1 si no se conoce. Consulte las notas 1 y 2.	
AVERAGE_SEQUENCE_PAGES	DOUBLE		Promedio de páginas del índice accesibles en secuencia (es decir, el número de páginas del índice que la captación previa detectaría que forman una secuencia); –1 si no se conoce. Consulte la nota 2.	
AVERAGE_SEQUENCE_FETCH_PAGES	DOUBLE		Promedio de páginas de la tabla accesibles en secuencia (es decir, el número de páginas de la tabla que la captación previa detectaría que forman una secuencia) al captar utilizando el índice; –1 si no se conoce. Consulte las notas 1 y 2.	

Notas:

1. Cuando se utilizan espacios de tabla DMS, no puede calcularse esta estadística.
2. No se recopilan estadísticas de captación previa durante una operación LOAD...STATISTICS YES o CREATE INDEX...COLLECT STATISTICS ni cuando el parámetro de configuración *seqdetect* está desactivado.

SYSSTAT.ROUTINES

Contiene una fila para cada función definida por el usuario (escalar, de tabla o de fuente), método generado por el sistema, método definido por el usuario o procedimiento. No incluye las funciones incorporadas. (Esta vista de catálogo reemplaza a SYSSTAT.FUNCTIONS. La otra vista existe pero permanecen igual que en DB2 Versión 7.1.)

Tabla 122. Vista de catálogo SYSSTAT.ROUTINES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción	Actualizable
ROUTINESCHEMA	VARCHAR(128)		Nombre de rutina calificado.	
ROUTINENAME	VARCHAR(128)			
ROUTINETYPE	CHAR(1)		F = Función M = Método P = Procedimiento.	
SPECIFICNAME	VARCHAR(128)		El nombre de la instancia de rutina (puede ser generado por el sistema).	
IOS_PER_INVOC	DOUBLE		El número estimado de operaciones de E/S por invocación; –1 si no se conoce (0 valor por omisión). Esta columna sólo puede actualizarse con –1 ó >= 0 (cero).	Sí
INSTS_PER_INVOC	DOUBLE		El número estimado de instrucciones por invocación; –1 si no se conoce (450 por omisión). Esta columna sólo puede actualizarse con –1 ó >= 0 (cero).	Sí
IOS_PER_ARGBYTE	DOUBLE		El número estimado de operaciones de E/S por byte de argumento de entrada; –1 si no se conoce (0 por omisión). Esta columna sólo puede actualizarse con –1 ó >= 0 (cero).	Sí
INSTS_PER_ARGBYTE	DOUBLE		El número estimado de instrucciones por byte de argumento de entrada; –1 si no se conoce (0 por omisión). Esta columna sólo puede actualizarse con –1 ó >= 0 (cero).	Sí
PERCENT_ARGBYTES	SMALLINT		El porcentaje medio estimado de bytes de argumento de entrada que la rutina leerá realmente; –1 si no se conoce (100 por omisión). Esta columna sólo puede actualizarse con –1 o un número entre 0 (cero) y 100.	Sí
INITIAL_IOS	DOUBLE		El número estimado de operaciones de E/S realizadas la primera/última vez que se invoca la rutina; –1 si no se conoce (0 por omisión). Esta columna sólo puede actualizarse con –1 ó >= 0 (cero).	Sí
INITIAL_INSTS	DOUBLE		El número estimado de instrucciones ejecutadas la primera/última vez que se invoca la rutina; –1 si no se conoce (0 por omisión). Esta columna sólo puede actualizarse con –1 ó >= 0 (cero).	Sí

Tabla 122. Vista de catálogo SYSSTAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción	Actualizable
CARDINALITY	BIGINT		La cardinalidad predicha de una función de tabla; –1 si no se conoce o si la rutina no es una función de tabla. Esta columna sólo puede actualizarse con –1 ó >= 0 (cero).	Sí
SELECTIVITY	DOUBLE		Utilizado para predicados definidos por el usuario; –1 si no hay ningún predicado definido por el usuario. Consulte la Nota 1.	

Notas:

1. Esta columna se establece en –1 al migrar desde la Versión 5.2 a la Versión 8.1 de DB2 en los catálogos del sistema para todas las funciones definidas por el usuario. Para un predicado definido por el usuario, la selectividad es –1 en el catálogo del sistema. En este caso, el valor de selectividad utilizado por el optimizador es 0.01.

SYSSTAT.TABLES

Contiene una fila para cada tabla *base*. Por lo tanto, las vistas o seudónimos no se incluyen. Para las tablas con tipo, sólo se incluye la tabla raíz de una jerarquía de tablas en esta vista. Las estadísticas no se registran para columnas heredadas de tablas con tipo. El valor CARD se aplica a la tabla raíz solamente mientras las otras estadísticas se apliquen a toda la jerarquía de tablas.

Tabla 123. Vista de catálogo SYSSTAT.TABLES

Nombre de columna	Tipo de datos	Posibilidad de nulos	Descripción	Actualizable
TABSCHEMA	VARCHAR(128)		Nombre calificado de la tabla.	
TABNAME	VARCHAR(128)			
CARD	BIGINT		El número total de filas en la tabla; –1 si no se reúnen estadísticas. Una actualización a CARD para una tabla no debe intentar asignarle un valor menor que el valor COLCARD de cualquiera de las columnas de dicha tabla. No puede cambiarse un valor de –2 y un valor de columna no puede establecerse directamente en –2. Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó >= 0 (cero)	Sí
NPAGES	INTEGER		El número total de páginas en las que las filas de la tabla existen; –1 si no se reúnen estadísticas; –2 para subtablas y tablas-H. No puede cambiarse un valor de –2 y un valor de columna no puede establecerse directamente en –2. Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó >= 0 (cero)	Sí
FPAGES	INTEGER		El número total de páginas en el archivo; –1 si no se reúnen estadísticas y –2 para subtablas y tablas-H. No puede cambiarse un valor de –2 y un valor de columna no puede establecerse directamente en –2. Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó > 0 (cero)	Sí
OVERFLOW	INTEGER		El número total de registros de desbordamiento de la tabla; –1 si no se reúnen estadísticas y –2 para subtablas y tablas-H. No puede cambiarse un valor de –2 y un valor de columna no puede establecerse directamente en –2. Esta columna sólo puede actualizarse con los valores siguientes: • –1 ó >= 0 (cero)	Sí
CLUSTERED	CHAR(1)	Sí	Y = La tabla está en un clúster de varias dimensiones (aunque sólo según una dimensión). Nulo = La tabla no está en un clúster.	
ACTIVE_ BLOCKS	INTEGER		El número total de páginas de bloques en uso en una tabla con clústeres de varias dimensiones (MDC); –1 si no se reúnen estadísticas.	Sí

Apéndice E. Sistemas federados

Tipos de servidores válidos en sentencias de SQL

Los tipos de servidor indican la clase de fuente de datos que la definición del servidor representa. Los tipos de servidores varían según el proveedor, la finalidad y el sistema operativo. Los valores soportados varían según el reiniciador utilizado.

Para la mayor parte de las fuentes de datos, debe especificar un tipo de servidor válido en la sentencia CREATE SERVER.

Reiniciador BioRS

Fuentes de datos BioRS.

Tipo de servidor	Fuente de datos
No es obligatorio en la sentencia CREATE SERVER.	BioRS

Reiniciador BLAST

Fuentes de datos BLAST a las que el daemon BLAST proporciona soporte.

Tipo de servidor	Fuente de datos
BLASTN	Búsquedas de BLAST en las que una secuencia de nucleótidos se compara con el contenido de una base de datos de secuencias de nucleótidos para hallar secuencias con regiones homólogas a las regiones de la secuencia original.
BLASTP	Búsquedas de BLAST en las que una secuencia de aminoácidos se compara con el contenido de una base de datos de secuencias de aminoácidos para hallar secuencias con regiones homólogas a las regiones de la secuencia original.
BLASTX	Búsquedas de BLAST en las que una secuencia de nucleótidos se compara con el contenido de una base de datos de secuencias de aminoácidos para hallar secuencias con regiones homólogas a las regiones de la secuencia original.
TBLASTN	Búsquedas de BLAST en las que una secuencia de aminoácidos se compara con el contenido de una base de datos de secuencias de nucleótidos para hallar secuencias con regiones homólogas a las regiones de la secuencia original.

Tipo de servidor	Fuente de datos
TBLASTX	Búsquedas de BLAST en las que una secuencia de nucleótidos se compara con el contenido de una base de datos de secuencias de nucleótidos para hallar secuencias con regiones homólogas a las regiones de la secuencia original.

Reiniciador CTLIB

Fuentes de datos Sybase a las que el software cliente CTLIB proporciona soporte.

Tipo de servidor	Fuente de datos
SYBASE	Sybase

Reiniciador Documentum

Fuentes de datos Documentum a las que la API/biblioteca del cliente Documentum proporciona soporte.

Tipo de servidor	Fuente de datos
DCTM	Documentum

Reiniciador DRDA

Fuentes de datos de la familia DB2

Tabla 124. DB2 para Linux, UNIX y Windows

Tipo de servidor	Fuente de datos
DB2/UDB	IBM DB2 Universal Database
DB2/6000	IBM DB2 para AIX
DB2/AIX	IBM DB2 para AIX
DB2/HPUX	IBM DB2 para HP-UX
DB2/HP	IBM DB2 para HP-UX
DB2/NT	IBM DB2 para Windows NT
DB2/EEE	IBM DB2 Enterprise-Extended Edition
DB2/SUN	IBM DB2 para Solaris
DB2/PE	IBM DB2 para Personal Edition
DB2/2	IBM DB2 para OS/2
DB2/LINUX	IBM DB2 para Linux
DB2/PTX	IBM DB2 para NUMA-Q
DB2/SCO	IBM DB2 para SCO Unixware

Tabla 125. DB2 para iSeries (y AS/400)

Tipo de servidor	Fuente de datos
DB2/400	IBM DB2 para iSeries y AS/400

Tabla 126. DB2 para z/OS y OS/390

Tipo de servidor	Fuente de datos
DB2/ZOS	IBM DB2 para z/OS
DB2/390	IBM DB2 para OS/390
DB2/MVS	IBM DB2 para MVS

Tabla 127. DB2 Server para VM y VSE

Tipo de servidor	Fuente de datos
DB2/VM	IBM DB2 para VM
DB2/VSE	IBM DB2 para VSE
SQL/DS	IBM SQL/DS

Reiniciador Entrez

Fuentes de datos Entrez.

Tipo de servidor	Fuente de datos
NUCLEOTIDE	Entrez
PUBMED	Entrez

Reiniciador Excel

Fuentes de datos Excel a las que Microsoft Excel 97, 2000, y 2002 proporciona soporte.

Tipo de servidor	Fuente de datos
No es obligatorio en la sentencia CREATE SERVER.	Microsoft Excel

Reiniciador Extended Search

Fuentes de datos Extended Search a las que la biblioteca del cliente Extended Search proporciona soporte.

Tipo de servidor	Fuente de datos
No es obligatorio en la sentencia CREATE SERVER.	IBM Lotus Extended Search

Reiniciador HMMER

Fuentes de datos HMMER a las que el daemon HMMER proporciona soporte.

Tipo de servidor	Fuente de datos
PFAM	HMMER
SEARCH	HMMER

Reiniciador Informix

Fuentes de datos Informix a las que el software SDK cliente de Informix proporciona soporte.

Tipo de servidor	Fuente de datos
INFORMIX	Informix

Reiniciador MSSQLODBC3

Fuentes de datos de Microsoft SQL Server a las que el controlador DataDirect Connect ODBC 3.6 o el controlador ODBC 3.0 (o superior) proporciona soporte.

Tipo de servidor	Fuente de datos
MSSQLSERVER	Microsoft SQL Server

Reiniciador NET8

Fuentes de datos Oracle a las que el software cliente NET8 de Oracle proporciona soporte.

Tipo de servidor	Fuente de datos
ORACLE	Oracle Versión 8.0. o posterior

Reiniciador ODBC

Fuentes de datos ODBC a las que el controlador ODBC 3.x proporciona soporte.

Tipo de servidor	Fuente de datos
ODBC	ODBC

Reiniciador OLE DB

Proveedores OLE DB compatibles con Microsoft OLE DB 2.0 o posterior.

Tipo de servidor	Fuente de datos
No es obligatorio en la sentencia CREATE SERVER.	Cualquier proveedor OLE DB

Reiniciador de archivos estructurados-tabla

Fuentes de datos de archivos estructurados-tabla.

Tipo de servidor	Fuente de datos
No es obligatorio en la sentencia CREATE SERVER.	Archivos estructurados-tabla

Reiniciador Teradata

Fuentes de datos Teradata a las que el software cliente de Teradata V2R3, V2R4 y V2R5 proporciona soporte.

Tipo de servidor	Fuente de datos
TERADATA	Teradata

Reiniciador de Servicios Web

Fuentes de datos Servicios Web.

Tipo de servidor	Fuente de datos
No es obligatorio en la sentencia CREATE SERVER.	Cualquier fuente de datos de Servicios Web.

Reiniciador de Integración comercial de WebSphere

Fuentes de datos de aplicaciones comerciales a las que el reiniciador de Integración comercial de WebSphere proporciona soporte.

Tipo de servidor	Fuente de datos
WBI	Integración comercial de WebSphere 2.2 o 2.3

Reiniciador XML

Fuentes de datos XML.

Tipo de servidor	Fuente de datos
No es obligatorio en la sentencia CREATE SERVER.	XML

Opciones de la columna de apodo para sistemas federados

Es posible especificar información de columna en las sentencias CREATE NICKNAME o ALTER NICKNAME utilizando los parámetros denominados opciones de columna de apodo.

La tabla siguiente muestra las opciones de la columna de apodo para cada fuente de datos.

Tabla 128. Opciones disponibles de la columna de apodo

Fuente de datos	ALL_VALUES	DEFAULT	DELIMITER	DOCUMENT	ESCAPE_INPUT	FOREIGN_KEY	INDEX	IS_REPEATING	NUMERIC_STRING	PRIMARY_KEY	REMOTE_NAME	SOAPACTIONCOLUMN	TEMPLATE	URLCOLUMN	VARCHAR_NO_TRAILING_ BLANKS	XPATH
BLAST		X	X				X									
DB2 Universal Database para iSeries									X							
DB2 Universal Database para z/OS y OS/390									X							
DB2 Universal Database para VM y VSE									X							
DB2 Universal Database para Linux, UNIX y Windows									X							
Documentum	X		X					X			X					
Informix									X							
Microsoft SQL Server									X							
ODBC									X							
OLE DB																
Oracle									X						X	
Sybase									X							
Archivos con estructura de tabla				X												
Teradata									X							
Integración comercial de WebSphere					X	X				X			X			X
Servicios Web					X	X				X		X	X	X		X
XML				X		X				X						X

Tabla 129. Opciones de columna y los valores de las mismas

Opción	Descripción y valores válidos	Valor por omisión
ALL_VALUES	Especifica que se devolverán todos los valores de un atributo repetitivo, separados mediante el delimitador especificado. Si falta esta opción o está establecida en N, sólo se devuelve el último valor de un atributo repetitivo. La opción ALL_VALUES sólo puede especificarse para columnas VARCHAR para las que la opción IS_REPEATING esté establecida en 'Y' (y no es válida cuando IS_REG_TABLE = 'Y').	
DEFAULT	Especifica un valor por omisión nuevo para las columnas de entrada fijas siguientes: • E_value • QueryStrands • GapAlign • NMisMatchPenalty • NMatchReward • Matrix • FilterSequence • NumberOfAlignments • GapCost • ExtendedGapCost • WordSize • ThresholdEx Este nuevo valor altera temporalmente los valores por omisión preestablecidos. El nuevo valor por omisión debe ser del mismo tipo que el valor indicado para una columna determinada.	
DELIMITER	Para Documentum: Especifica la serie de delimitador que se debe utilizar cuando varios valores de un atributo repetitivo están concatenados. El delimitador puede ser uno o varios caracteres. Esta opción sólo es válida para los atributos de objetos con el tipo de datos VARCHAR en los que la opción IS_REPEATING esté establecida en Y. Para BLAST: Los caracteres de delimitador que deben utilizarse para determinar el punto final de la información de la línea de definición para la columna en la que aparece esta opción. Si aparece más de un carácter en el valor de esta opción, la primera ocurrencia de cualquiera de los caracteres indica el final de la información de este campo. El valor por omisión es el final de la línea. Esta opción es obligatoria, a menos que desee que la última columna especificada contenga el resto de la línea de definición.	Para Documentum: El delimitador por omisión es una coma. Para BLAST: El delimitador por omisión es el final de la línea.

Tabla 129. Opciones de columna y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
DOCUMENT	Para los archivos estructurados en tabla: Especifica el tipo de archivo estructurado en tabla. Este reiniciador sólo proporciona soporte al valor FILE para esta opción. Sólo es posible especificar una columna por apodo con la opción DOCUMENT. La columna que está asociada con la opción DOCUMENT debe ser un tipo de datos VARCHAR o CHAR. Si se utiliza la opción de la columna de apodo DOCUMENT en lugar de la opción de apodo FILE_PATH, se proporcionará el archivo que se corresponda con este apodo durante la ejecución de la consulta. Si la opción DOCUMENT tiene el valor FILE, el valor que se proporciona cuando se ejecuta la consulta es la vía de acceso completa del archivo cuyo esquema coincida con la definición de apodo de este apodo. Para XML: Especifica que esta columna es una columna DOCUMENT. El valor de la columna DOCUMENT indica el tipo de fuente de datos XML que se proporciona al apodo cuando se ejecuta la consulta. Esta opción sólo se acepta para las columnas del apodo raíz (el apodo que identifica los elementos del nivel superior del documento XML). Sólo es posible especificar una columna por apodo con la opción DOCUMENT. La columna que está asociada a la opción DOCUMENT debe ser un tipo de datos VARCHAR. Si se utiliza una opción de columna DOCUMENT en lugar de la opción de apodo FILE_PATH o DIRECTORY_PATH, el documento que se corresponde con este apodo se proporciona cuando se ejecuta la consulta. Los valores válidos para la opción DOCUMENT son: **FILE** Especifica que el valor de la columna de apodo está enlazado con el nombre de la vía de acceso de un archivo. Los datos de este archivo se proporcionan cuando se ejecuta la consulta. **DIRECTORY** Especifica que el valor de la columna de apodo está enlazado con el nombre de la vía de acceso de un directorio que contiene varios archivos de datos XML. Los datos XML de los distintos archivos se proporcionan cuando se ejecuta la consulta. Los datos se encuentran en archivos XML en la vía de acceso del directorio especificado. El reiniciador de XML sólo utiliza los archivos con extensión .xml que se encuentren en el directorio especificado. El reiniciador de XML hace caso omiso del resto de archivos de este directorio. **URI** Especifica que el valor de la columna de apodo está enlazado con el nombre de la vía de acceso de un archivo XML remoto al que el URI hace referencia. La dirección del URI indica la ubicación remota de este archivo XML en la Web. **COLUMN** Especifica que el documento XML está almacenado en una columna relacional.	

Tabla 129. Opciones de columna y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
ELEMENT_NAME	Especifica el nombre de elemento de BioRS. La sensibilidad a las mayúsculas y minúsculas de este nombre depende del servidor BioRS y del valor de la opción del servidor CASE_SENSITIVE. Sólo se debe especificar el nombre de elemento de BioRS si es distinto del nombre de columna.	
ESCAPE_INPUT	Especifica si los caracteres especiales de XML se sustituyen o no se sustituyen en los valores de salida de XML. Este opción se utiliza para incluir fragmentos de XML en forma de entrada, como, por ejemplo, fragmentos de XML con elementos repetitivos. La opción de columna TEMPLATE debe estar definida en las columnas que utilicen la opción de columna ESCAPE_INPUT. El tipo de datos de columna debe ser VARCHAR o CHAR. Los valores válidos son: Y Si la salida de XML contiene caracteres especiales, éstos se sustituyen por los caracteres adicionales que XML utiliza para representar los caracteres de entrada. N Los caracteres de entrada se mantienen exactamente tal como aparecen.	Y
FOREIGN_KEY	Indica que este apodo es un apodo hijo y especifica el nombre del apodo padre correspondiente. Un apodo puede tener una opción de columna FOREIGN_KEY como máximo. El valor de esta opción es sensible a las mayúsculas y minúsculas. La tabla indicada por esta opción contiene una clave generada por el reiniciador. La opción XPATH no puede especificarse para esta columna. La columna sólo puede utilizarse para unir apodos padre con apodos hijo. Una sentencia CREATE NICKNAME con una opción FOREIGN_KEY fallará si el apodo padre presenta un nombre de esquema distinto. A menos que el apodo al que la cláusula FOREIGN_KEY hace referencia se haya definido de forma explícita en minúsculas o combinando caracteres en mayúsculas y minúsculas escribiéndolo entre comillas en la sentencia CREATE NICKNAME correspondiente, cuando haga referencia a este apodo en la cláusula FOREIGN_KEY, el apodo debe especificarse en mayúsculas. Cuando esta opción está establecida en una columna, no puede establecerse ninguna otra opción en la columna.	
INDEX	El número de orden de la columna en la que aparece esta opción en el grupo de columnas de la línea de definición. Esta opción es obligatoria.	
IS_INDEXED	Indica si la columna correspondiente está indexada(si es posible hacer referencia a la columna en un predicado). Los valores válidos son Y y N. El valor Y sólo puede especificarse para las columnas en las que el servidor BioRS haya indexado el elemento correspondiente.	Cuando se crea un apodo, esta opción se añade de forma automática con el valor Y a todas las columnas correspondientes a un elemento indexado por BioRS.

Tabla 129. Opciones de columna y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
IS_REPEATING	Indica si la columna tiene varios valores. Los valores válidos son Y y N. Sólo se devuelve el último valor para: • los atributos repetitivos que no son VARCHAR • las columnas VARCHAR cuando se ha especificado ALL_VALUES 'N' Para superar esta limitación, es posible crear una definición doble para la columna de atributo repetitivo.	N
NUMERIC_STRING	Especifica si una columna contiene series de caracteres numéricos. Y Esta columna contiene series de caracteres numéricos '0', '1', '2', ... '9'. No contiene blancos. Si esta columna sólo contiene series numéricas seguidas de blancos de cola, no debe especificarse Y. Cuando se establece NUMERIC_STRING en Y para una columna, se informa al optimizador de que esta columna no contiene blancos que puedan interferir en la clasificación de los datos de la columna. Esta opción se utiliza cuando el orden de clasificación de una fuente de datos es diferente del orden de clasificación que el servidor federado utiliza. Las columnas que utilizan esta opción no se excluyen de la evaluación remota debido a un orden de clasificación diferente. N Esta columna o no es una columna de serie numérica o es una columna de serie numérica que contiene blancos.	N
PRIMARY_KEY	Indica que este apodo es un apodo padre. El tipo de datos de columna debe ser VARCHAR(16). Un apodo puede tener una opción de columna PRIMARY_KEY como máximo. YES es el único valor válido. La columna indicada por esta opción contiene una clave generada por el reiniciador. La opción XPATH no puede especificarse para esta columna. La columna sólo puede utilizarse para unir apodos padre con apodos hijo. Cuando esta opción está establecida en una columna, no puede establecerse ninguna otra opción en la columna.	
REFERENCED_OBJECT	Esta opción sólo es válida para las columnas cuyo tipo de datos BioRS sea Reference. Esta opción especifica el nombre del banco de datos de BioRS al que la columna actual hace referencia. La sensibilidad a las mayúsculas y minúsculas de este nombre depende del servidor BioRS y del valor de la opción del servidor CASE_SENSITIVE.	
REMOTE_NAME	Especifica el nombre del atributo o la columna de Documentum correspondiente. Esta opción correlaciona nombres de atributos o de columnas remotos con nombres de columnas de UDB locales.	El nombre de la columna de DB2 UDB.

Tabla 129. Opciones de columna y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
SOAPACTIONCOLUMN	Una columna para especificar de forma dinámica el atributo URI SOAPACTION del formato del lenguaje de descripción de servicios Web (WSDL). Esta opción sólo se especifica en el apodo raíz. Cuando esta opción está establecida en una columna, no puede establecerse ninguna otra opción en la columna.	
TEMPLATE	El fragmento de plantilla de columna que debe utilizarse para construir el documento de entrada de XML. El fragmento debe cumplir la sintaxis especificada para las plantillas.	
URLCOLUMN	Una columna para especificar de forma dinámica el URL para el punto final de servicios Web cuando se ejecuta una consulta. Esta opción sólo se especifica en el apodo raíz. Cuando esta opción está establecida en una columna, no puede establecerse ninguna otra opción en la columna.	
VARCHAR_NO_ TRAILING_BLANKS	Esta opción se aplica a las fuentes de datos que tienen tipos de datos de caracteres variables que no rellenan la longitud con blancos de cola durante la comparación. Algunas fuentes de datos como, por ejemplo, Oracle, carecen de una semántica de comparación con caracteres de relleno de blancos que devuelvan el mismo resultado que la semántica de comparación de DB2 UDB para Linux, UNIX y Windows. Defina esta opción cuando desee que sólo se aplique a una columna VARCHAR o VARCHAR2 determinada de un objeto de fuente de datos. Y — No aparecen blancos de cola en estas columnas VARCHAR o la fuente de datos presenta una semántica de comparación de caracteres de relleno de blancos similar a la semántica del servidor federado. El servidor federado envía las operaciones de comparación de caracteres a la fuente de datos para su proceso. N — Aparecen blancos de cola en estas columnas VARCHAR y la fuente de datos presenta una semántica de comparación de caracteres de relleno de blancos diferente de la del servidor federado. El servidor federado procesa las operaciones de comparación de caracteres si no es posible realizar una compensación con una semántica equivalente. Por ejemplo, vuelve a escribir el predicado.	N para las fuentes de datos afectadas
XPATH	Especifica la expresión XPath en el documento XML que contiene los datos correspondientes a esta columna. El reiniciador evalúa la expresión XPath después de que la sentencia CREATE NICKNAME se aplique a esta expresión XPath desde esta opción de apodo XPATH.	

Conceptos relacionados:

- "Pushdown analysis" en la publicación *Federated Systems Guide*

Tareas relacionadas:

- "Global optimization" en la publicación *Federated Systems Guide*

Opciones de correlación de funciones para sistemas federados

DB2 Information Integrator proporciona correlaciones por omisión entre funciones incorporadas de fuente de datos existentes y funciones incorporadas de DB2. Para la mayor parte de fuentes de datos, las correlaciones de funciones por omisión se encuentran en los reiniciadores. Para utilizar una función de fuente de datos que el servidor federado no reconoce, el usuario debe crear una correlación de funciones entre una función de fuente de datos y una función complementaria situada en la base de datos federada.

La finalidad principal de las opciones de correlación de funciones es proporcionar información acerca del coste potencial de ejecutar una función de fuente de datos en la fuente de datos. El análisis de bajada determina si una función de la fuente de datos es capaz de ejecutar una función situada en una consulta. El optimizador de consultas decide si bajar el proceso de la función a la fuente de datos es la alternativa menos costosa.

La información de estadísticas proporcionada en la definición de correlación de funciones ayuda al optimizador de consultas a comparar el coste estimado de la ejecución de la función en la fuente de datos con el coste estimado de ejecutar la función en DB2.

Tabla 130. Opciones de correlación de funciones y los valores de las mismas

Opción	Valores válidos	Valor por omisión
DISABLE	Inhabilita una correlación de funciones por omisión. Los valores válidos son 'Y' y 'N'.	'N'
INITIAL_INSTS	El número estimado de instrucciones procesadas la primera y la última vez que se ha invocado la función de la fuente de datos.	'0'
INITIAL_IOS	Número estimado de E/S realizadas la primera y la última vez que se ha invocado la función de la fuente de datos.	'0'
IOS_PER_ARGBYTE	Número estimado de E/S expendidas por cada byte del conjunto de argumentos que se pasa a la función de la fuente de datos.	'0'
IOS_PER_INVOC	Número estimado de E/S por invocación de una función de fuente de datos.	'0'
INSTS_PER_ARGBYTE	Número estimado de instrucciones procesadas por cada byte del conjunto de argumentos que se pasa a la función de fuente de datos.	'0'
INSTS_PER_INVOC	Número estimado de instrucciones procesadas por invocación de la función de fuente de datos.	'450'
PERCENT_ARGBYTES	Porcentaje medio estimado de bytes de argumento de entrada que la función de fuente de datos leerá realmente.	'100'
REMOTE_NAME	Nombre de la función de fuente de datos.	nombre local

Opciones del servidor para sistemas federados

Las opciones del servidor se utilizan para describir un servidor de fuentes de datos. Las opciones del servidor especifican información sobre la integridad, la ubicación, la seguridad y el rendimiento. Algunas opciones del servidor están disponibles para todas las fuentes de datos y otras opciones del servidor son específicas para la fuente de datos.

Las opciones del servidor federadas comunes para fuentes de datos relacionales son las siguientes:

- Opciones de compatibilidad. COLLATING_SEQUENCE, IGNORE_UDT
- Opciones de integridad de los datos. IUD_APP_SVPT_ENFORCE
- Opciones de fecha y hora. DATEFORMAT, TIMEFORMAT, TIMESTAMPFORMAT
- Opciones de ubicación. CONNECTSTRING, DBNAME, IFILE
- Opciones de seguridad. FOLD_ID, FOLD_PW, INFORMIX_LOCK_MODE
- Opciones de rendimiento. COMM_RATE, CPU_RATIO, DB2_MAXIMAL_PUSHDOWN, IO_RATIO, LOGIN_TIMEOUT, PACKET_SIZE, PLAN_HINTS, PUSHDOWN, TIMEOUT, VARCHAR_NO_TRAILING_BLANKS

La tabla siguiente muestra las opciones del servidor para la definición del servidor aplicables a todas las fuentes de datos relacionales:

Tabla 131. Opciones del servidor para las fuentes de datos relacionales

Fuente de datos	CODEPAGE	COLLATING_SEQUENCE	COMM_RATE	CONNECTSTRING	CPU_RATIO	DATEFORMAT	DB2_MAXIMAL_PUSHDOWN	DBNAME	FOLD_ID	FOLD_PW	IFILE	INFORMIX_LOCK_MODE	IO_RATIO	IUD_APP_SVPT_ENFORCE	LOGIN_TIMEOUT	NODE	PACKET_SIZE	PASSWORD	PLAN_HINTS	PUSHDOWN	TIMEOUT	TIMEFORMAT	TIMESTAMPFORMAT	VARCHAR_NO_TRAILING_BLANKS
DB2 UDB para iSeries		X	X		X		X	X	X	X			X	X				X		X				X
DB2 UDB para z/OS y OS/390		X	X		X		X	X	X	X			X	X				X		X				X
DB2 para VM y VSE		X	X		X		X	X	X	X			X	X				X		X				X
DB2 UDB para Linux, UNIX y Windows		X	X		X		X	X	X	X			X	X				X		X				X
Informix		X	X		X		X	X	X	X		X	X	X		X		X		X				
Microsoft SQL Server	X	X	X		X		X	X	X	X			X	X		X		X		X				

Tabla 131. Opciones del servidor para las fuentes de datos relacionales (continuación)

Fuente de datos	CODEPAGE	COLLATING_SEQUENCE	COMM_RATE	CONNECTSTRING	CPU_RATIO	DATEFORMAT	DB2_MAXIMAL_PUSHDOWN	DBNAME	FOLD_ID	FOLD_PW	IFILE	INFORMIX_LOCK_MODE	IO_RATIO	IUD_APP_SVPT_ENFORCE	LOGIN_TIMEOUT	NODE	PACKET_SIZE	PASSWORD	PLAN_HINTS	PUSHDOWN	TIMEOUT	TIMEFORMAT	TIMESTAMPFORMAT	VARCHAR_NO_TRAILING_BLANKS
ODBC	X	X	X		X	X	X	X	X	X			X	X		X		X		X		X	X	X
OLE DB		X		X																				
Oracle		X	X		X		X		X	X			X			X		X	X	X				X
Sybase		X	X		X		X	X	X	X	X		X		X	X	X	X	X	X	X			
Teradata		X	X		X		X						X	X		X				X				

La tabla siguiente muestra las opciones del servidor para la definición del servidor aplicables a las fuentes de datos no relacionales, a excepción de Integración comercial de WebSphere: Las opciones del servidor para la definición del servidor de Integración comercial de WebSphere aparecen en la Tabla 133 en la página 685.

Tabla 132. Opciones del servidor para las fuentes de datos no relacionales.

Fuente de datos	CASE_SENSITIVE	CONTENT_DIR	DAEMON_PORT	ES_HOST	ES_PORT	ES_TRACING	ES_TRACELEVEL	ES_TRACEFILENAME	HMMPFAM_OPTIONS	HMMSEARCH_OPTIONS	MAX_ROWS	NODE	OS_TYPE	PORT	PROCESSORS	PROXU_AUTHID	PROXY_PASSWORD	PROXY_SERVER_NAME	PROXY_SERVER_PORT	PROXY_TYPE	RDBMS_TYPE	SOCKET_TIMEOUT	TIMEOUT	TRANSACTIONS	USE_CLOB_SEQUENCE
BioRS	X											X		X									X		
BLAST			X									X													X
Documentum		X										X	X								X			X	
Entrez											X					X	X	X	X	X		X			
Excel																									
Extended Search				X	X	X	X	X																	
HMMER			X						X	X		X			X										X
Archivos estructurados-tabla																									
Servicios Web																									
XML																X	X	X	X	X		X			

La tabla siguiente muestra las opciones del servidor para la definición del servidor aplicables a las fuentes de datos de Integración comercial de WebSphere:

Tabla 133. Opciones del servidor para las fuentes de datos de Integración comercial de WebSphere

Fuente de datos	APP_TYPE	FAULT_QUEUE	MQ_CONN_NAME	MQ_MANAGER	MQ_RESPONSE_TIMEOUT	MQ_SVRCONN_CHANNELNAME	REQUEST_QUEUE	RESPONSE_QUEUE
Integración comercial de WebSphere	X	X	X	X	X	X	X	X

La tabla siguiente describe todas las opciones del servidor y muestra los valores válidos y los valores por omisión.

Tabla 134. Opciones de servidor y los valores de las mismas

Opción	Descripción y valores válidos	Valor por omisión
APP_TYPE	El tipo de la aplicación remota. Los valores válidos son 'PSOFT', 'SAP' y 'SIEBEL'. Esta opción es obligatoria.	Ninguno.

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
CASE_SENSITIVE	Especifica si el servidor BioRS trata los nombres distinguiendo entre mayúsculas y minúsculas. Los valores válidos son Y o N. 'Y' El servidor BioRS trata los nombre distinguiendo entre mayúsculas y minúsculas. 'N' El servidor BioRS no trata los nombres distinguiendo entre mayúsculas y minúsculas. En el producto BioRS, un parámetro de configuración controla la sensibilidad a las mayúsculas y minúsculas de los datos almacenados en el servidor BioRS. La opción CASE_SENSITIVE es el parámetro de DB2 Information Integrator complementario al parámetro de configuración del sistema. Es necesario sincronizar los valores de sensibilidad a mayúsculas y minúsculas del servidor BioRS del sistema BioRS y de DB2 Information Integrator. Si los valores de configuración de la sensibilidad a mayúsculas y minúsculas no se mantienen sincronizados entre BioRS y DB2 Information Integrator, se producirán errores al intentar acceder a los datos de BioRS a través de DB2 Information Integrator. No es posible modificar ni suprimir la opción CASE_SENSITIVE después de crear un servidor BioRS nuevo en DB2 Information Integrator. Si es necesario modificar la opción CASE_SENSITIVE, debe descartar todo el servidor y volver a crearlo. Si se descarta el servidor BioRS, también es necesario volver a crear todos los apodos de BioRS correspondientes. DB2 Information Integrator descarta automáticamente todos los apodos correspondientes a un servidor descartado.	Y
CODEPAGE	Especifica el identificador de la página de códigos de DB2 correspondiente al juego de caracteres codificados de la configuración del cliente de la fuente de datos. La página de códigos del cliente debe especificarse si la página de códigos del cliente y la página de códigos de la base de datos federada no coinciden. Para las fuentes de datos que proporcionan soporte a Unicode, la opción CODEPAGE puede establecerse en el identificador de la página de códigos de DB2 correspondiente a la codificación Unicode del cliente de la fuente de datos a la que se proporcione soporte.	En sistemas UNIX o Windows con una base de datos federada que no sea Unicode: La página de códigos de la base de datos federada. En sistemas UNIX con una base de datos federada Unicode: 1208 En sistemas Windows con una base de datos federada Unicode: 1202

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
COLLATING_ SEQUENCE	Especifica si la fuente de datos utiliza el mismo orden de clasificación por omisión que la base de datos federada, basándose en el conjunto de códigos NLS y la información de país/región. 'Y' La fuente de datos utiliza el mismo orden de clasificación que la base de datos federada de DB2. 'N' La fuente de datos tiene un orden de clasificación distinto al de la base de datos federada de DB2. 'I' La fuente de datos tiene un orden de clasificación distinto al de la base de datos federada de DB2 y el orden de clasificación de la fuente de datos no es sensible a mayúsculas y minúsculas (por ejemplo, se considera que 'STEWART' y 'StewART' son iguales).	'N'
COMM_RATE	Especifica la velocidad de comunicación entre el servidor federado y el servidor de la fuente de datos. Expresada en megabytes por segundo. Los valores válidos son mayores que 0 y menores que $1x10^{23}$. Los valores pueden expresarse en cualquier notación REAL válida.	'2'
CONTENT_DIR	Especifica el nombre del directorio raíz de acceso local para almacenar los archivos de contenido recuperados por las pseudocolumnas GET_FILE, GET_FILE_DEL, GET_RENDITION y GET_RENDITION_DEL. Todos los usuarios que puedan utilizar estas pseudocolumnas debe poder escribir en él.	En sistemas UNIX: '/tmp' En sistemas Windows: 'C:\temp'
CONNECTSTRING	Especifica las propiedades de inicialización necesarias para conectarse con un proveedor OLE DB.	Ninguno.
CPU_RATIO	Indica la diferencia de velocidad en que ejecuta la CPU de una fuente de datos con respecto a la CPU del servidor federado. Los valores válidos son mayores que 0 y menores que $1x10^{23}$. Los valores pueden expresarse en cualquier notación REAL válida. Un valor de 1 indica que la velocidad de la CPU de la DB2 federada y la velocidad de la CPU de la fuente de datos tienen la misma velocidad de la CPU, una proporción 1:1. Un valor de .5 indica que la CPU de la DB2 federada es un 50% más lenta que la velocidad de la CPU de la fuente de datos. Un valor de 2 indica que la velocidad de la CPU de la DB2 federada es el doble de rápida que la velocidad de la CPU de la fuente de datos.	'1,0'

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
DATEFORMAT	El formato de fecha utilizado por la fuente de datos. El formato debe introducirse utilizando 'DD', 'MM' e 'YY' o 'YYYY' para representar el formato numérico de la fecha. También es necesario especificar el delimitador como, por ejemplo, un espacio o una coma. Así pues, para representar el formato de fecha de '01-01-2003', debe utilizarse 'DD-MM-YYYY'. Este campo puede contener nulos.	Ninguno.
DAEMON_PORT	Especifica el número de puerto en el que el daemon escuchará las peticiones de trabajos de BLAST o HMMER. El número de puerto debe ser el mismo que el especificado en la opción DAEMON_PORT del archivo de configuración del daemon.	BLAST: '4007'; HMMER: '4098'
DB2_MAXIMAL_ PUSHDOWN	Especifica el criterio principal que el optimizador de consultas utiliza al seleccionar un plan de acceso. El optimizador de consultas puede seleccionar los planes de acceso en función del coste o en función del requisito del usuario de que las fuentes de datos remotas realicen todo el proceso posible de las consultas. 'Y' El optimizador de consultas selecciona un plan de acceso que baja más operaciones de consulta a la fuente de datos que otros planes. Cuando varios planes de acceso proporcionan la misma cantidad de bajada, el optimizador de consultas selecciona el plan con el menor coste. Si una tabla de consulta materializada (MQT) del servidor federado puede procesar una parte o la totalidad de la consulta, es posible utilizar un plan de acceso que incluya la tabla de consulta materializada. La base de datos federada no baja las consultas que generan un producto cartesiano. 'N' El optimizador de consultas selecciona un plan de acceso en función del coste.	'N'
DBNAME	Nombre de la base de datos de la fuente de datos a la que desea que el servidor federado acceda. Para una base de datos de DB2, este valor corresponde a una base de datos determinada para la conexión a la base de datos de DB2 remota inicial. Esta base de datos determinada es el alias de base de datos utilizado para la base de datos de DB2 remota que se haya catalogado en el servidor federado utilizando el mandato CATALOG DATABASE o el Asistente de configuración de DB2. No se aplica a las fuentes de datos Oracle porque las instancias de Oracle contienen una sola base de datos.	Ninguno.

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
ES_HOST	Especifica el nombre del sistema principal calificado al completo o la dirección IP del servidor Extended Search en el que desea realizar búsquedas. Esta opción es obligatoria.	Ninguno.
ES_PORT	Especifica el número de puerto en el que este servidor de Extended Search escucha las peticiones. Esta opción es opcional.	'6001'
ES_TRACING	Especifica si debe habilitarse el rastreo para los mensajes de error, los mensajes de aviso y los mensajes informativos generados por el servidor remoto de Extended Search. Los valores válidos son: 'OFF' No se anota cronológicamente ningún mensaje de rastreo. 'ON' Los mensajes de rastreo se anotan cronológicamente. Esta opción es opcional.	'OFF'

ES_TRACELEVEL	Si el rastreo está habilitado, esta opción especifica los tipos de mensajes que se escribirán en el archivo de anotaciones cronológicas. Es posible habilitar e inhabilitar los niveles de rastreo siguientes de forma independiente:
	'C' Mensajes de error graves.
	'N' Mensajes de error no graves.
	'W' Mensajes de aviso.
	'I' Mensajes informativos.
	Por ejemplo:
	`ES_TRACELEVEL 'W'`
	`ES_TRACELEVEL 'CN'`
	Esta opción es opcional.
		'C'

ES_TRACEFILENAME	Si el rastreo está habilitado, esta opción especifica el nombre del directorio y del archivo donde se escribirán los mensajes. Esta opción es opcional.	En los sistemas operativos UNIX: $INSTHOME/sqllib/log/ESWrapper.log. En los sistemas operativos Windows: %DB2TEMPDIR%\ESWrapper.log.
FAULT_QUEUE	El nombre de la cola de errores que entrega mensajes de error del adaptador al reiniciador. El nombre debe cumplir las especificaciones de los nombres de cola para WebSphere MQ. Se trata de una opción obligatoria.	Ninguno.

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
FOLD_ID (Consulte las notas 1 y 4 al final de esta tabla.)	Se aplica a los ID de usuario que el servidor federado envía al servidor de fuentes de datos para su autentificación. Los valores válidos son: 'U' El servidor federado convierte el ID de usuario a mayúsculas antes de enviarlo a la fuente de datos. Esta es una opción lógica para las fuentes de datos de la familia DB2 y Oracle. (Vea la nota 2 al final de esta tabla). 'N' El servidor federado no realiza ninguna acción en el ID de usuario antes de enviarlo a la fuente de datos. (Consulte la nota 2 al final de esta tabla.) 'L' El servidor federado convierte el ID de usuario a minúsculas antes de enviarlo a la fuente de datos. Si no se utiliza ninguno de estos valores, el servidor federado intenta enviar el ID de usuario a la fuente de datos en mayúsculas. Si el ID de usuario falla, el servidor intenta enviarlo en minúsculas.	Ninguno.
FOLD_PW (Consulte las notas 1, 3 y 4 al final de esta tabla.)	Se aplica a las contraseñas que el servidor federado envía a las fuentes de datos para que las autentifique. Los valores válidos son: 'U' El servidor federado convierte la contraseña a mayúsculas antes de enviarla a la fuente de datos. Esta es una opción lógica para las fuentes de datos de la familia DB2 y Oracle. 'N' El servidor federado no realiza ninguna acción en la contraseña antes de enviarla a la fuente de datos. 'L' El servidor federado convierte la contraseña a minúsculas antes de enviarla a la fuente de datos. Si no se utiliza ninguno de estos valores, el servidor federado intenta enviar la contraseña a la fuente de datos en mayúsculas. Si la contraseña falla, el servidor intenta enviarla en minúsculas.	Ninguno.
HMMPFAM_OPTIONS	Especifica opciones de hmmpfam como, por ejemplo, --null2, --pvm y --xnu que no tienen ningún nombre de columna correspondiente en una tabla de referencia que correlaciona las opciones con los nombres de las columnas. Por ejemplo: `HMMPFAM_OPTIONS '--xnu --pvm'` En este ejemplo, el daemon ejecuta el programa HMMPFAM con opciones a partir de la cláusula WHERE de la consulta más las opciones adicionales--xnu --pvm.	

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
HMMSEARCH_ OPTIONS	Permite al usuario proporcionar opciones de la línea de mandatos opcionales al mandato hmmsearch. Sólo es válido con el tipo SEARCH. Si desea obtener más información, consulte el manual del usuario de HMMER.	Ninguno.
IFILE	Especifica la vía de acceso y el nombre del archivo de interfaces Sybase Open Client. En los servidores federados de Windows NT, el valor por omisión es %DB2PATH%\interfaces. En los servidores federados UNIX, la vía de acceso por omisión y el valor del nombre es $ DB2INSTANCE/sqllib/interfaces.	Ninguno.
INFORMIX_LOCK_ MODE	Especifica la modalidad de bloqueo que debe definirse para una fuente de datos Informix. El reiniciador de Informix emite el mandato 'SET LOCK MODE' inmediatamente después de establecer la conexión con una fuente de datos Informix. Los valores válidos son: **'W'** Establece la modalidad de bloqueo de Informix en WAIT. Si el reiniciador intenta acceder a una tabla o a una fila bloqueada, Informix espera hasta que se libere el bloqueo. **'N'** Establece la modalidad de bloqueo de Informix en NOWAIT. Si el reiniciador intenta acceder a una tabla o a una fila bloqueada, Informix devuelve un error. **'n'** Establece la modalidad de bloqueo de Informix en WAIT *n* segundos. Si el reiniciador intenta acceder a una tabla o a una fila bloqueada y el bloqueo no se libera dentro del número de segundos especificado, Informix devuelve un error.	'W'
IO_RATIO	Indica la diferencia de velocidad en que se ejecuta el sistema de E/S de la fuente de datos con respecto al sistema de E/S del servidor federado. Los valores válidos son mayores que 0 y menores que $1x10^{23}$. Los valores pueden expresarse en cualquier notación REAL válida. Un valor de 1 indica que la velocidad de E/S de la DB2 federada y la velocidad de E/S de la fuente de datos tienen la misma velocidad de E/S, una proporción 1:1. Un valor de .5 indica que la velocidad de E/S de la DB2 federada es un 50% más lenta que la velocidad de E/S de la fuente de datos. Un valor de 2 indica que la velocidad de E/S de la DB2 federada es el doble de rápida que la velocidad de E/S de la fuente de datos.	'1,0'

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
IUD_APP_SVPT_ ENFORCE	Especifica si el sistema federado de DB2 debería imponer la detección o la creación de sentencias de puntos de salvar en la aplicación. Cuando se establece utilizando la sentencia SET SERVER OPTION, esta opción del servidor no afecta a las sentencias de SQL estático. 'Y' El servidor federado retrotrae las transacciones de inserción, actualización o supresión si se produce un error en una operación de inserción, actualización o supresión y la fuente de datos no impone sentencias de puntos de guardar en la aplicación. Se devuelve el código de error de SQL SQL1476N. 'N' El servidor federado no retrotraerá las transacciones cuando se detecte un error. Su aplicación debe gestionar la recuperación del error.	'Y'
LOGIN_TIMEOUT	Especifica el número de segundos que el servidor federado de DB2 debe esperar una respuesta de Sybase Open Client a la petición de inicio de sesión. Los valores por omisión son los mismos que para TIMEOUT.	'0'
MAX_ROWS	Especifica el número de filas que el servidor federado devuelve para una consulta que utilice el reiniciador de Entrez. Sólo se pueden especificar números positivos y el cero. Cuando se establece esta opción en cero, se permite que las consultas recuperen un número de filas ilimitado desde el sitio Web NCBI. Sin embargo, el establecimiento de la opción del servidor MAX_ROWS en cero o en un número muy elevado puede afectar el rendimiento de las consultas. La opción del servidor MAX_ROWS no es obligatoria.	Sistemas operativos Microsoft Windows: 2000 filas. Sistemas operativos basados en UNIX: 5000 filas.
MQ_CONN_NAME	El nombre del sistema principal o la dirección de red del sistema en el que se esté ejecutando el servidor Websphere MQ. Un ejemplo de un nombre de conexión es el siguiente: 9.30.76.151(1420) donde 1420 es el número de puerto. Si se excluye el número de puerto, se utiliza el valor por omisión 1414. Esta opción es opcional. Si se omite, se utiliza la variable de entorno MQSERVER (si está especificada en el archivo db2dj.ini) para seleccionar la definición del canal. Si no se establece MQSERVER, se utiliza la tabla del canal del cliente.	Si está especificada en el archivo db2dj.ini, el reiniciador utiliza la variable de entorno MQSERVER para seleccionar la definición del canal. Si no se establece la variable de entorno MQSERVER, el reiniciador utiliza la tabla del canal del cliente.
MQ_MANAGER	El nombre del gestor de WebSphere MQ. Cualquier nombre de gestor de WebSphere MQ válido. Esta opción es obligatoria.	Ninguno.

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
MQ_RESPONSE_ TIMEOUT	El tiempo que el reiniciador debe esperar un mensaje de respuesta de la cola de respuesta. El valor está en milésimas de segundo. Es posible especificar el valor especial -1 para indicar que no existe ningún tiempo de espera excedido. Esta opción es opcional.	10000
MQ_SVRCONN_ CHANNELNAME	El nombre del canal de conexión del servidor en el gestor de Websphere MQ al que reiniciador debe intentar conectarse. Este parámetro sólo puede especificarse si se especifica la opción del servidor MQ_CONN_NAME. Si se omite esta opción, se utiliza el canal de conexión del servidor por omisión, SYSTEM.DEF.SVRCONN.	SYSTEM.DEF.SVRCONN
NODE	Fuentes de datos relacionales: Nombre por el cual se define una fuente de datos como una instancia en su RDBMS. Documentum: Especifica el nombre real de Documentum Docbase. Esta opción es obligatoria. BLAST: Especifica el nombre del sistema principal del sistema en el que se está ejecutando el proceso del daemon BLAST. Esta opción es obligatoria. HMMER: Especifica el nombre del sistema principal del servidor en el que se ejecuta el proceso del daemon HMMER. Esta opción es obligatoria. BioRS: Especifica el nombre del sistema principal del sistema en el que está disponible la herramienta de consulta BioRS. Esta opción es opcional.	BioRS: *sistema_principal_local*
OS_TYPE	Especifica el sistema operativo del servidor Docbase. Los valores válidos son AIX, SOLARIS y WINDOWS. Esta opción es obligatoria.	Ninguno.
PACKET_SIZE	Especifica el tamaño del paquete del archivo de interfaces de Sybase en bytes. Si la fuente de datos no proporciona soporte al tamaño del paquete especificado, la conexión fallará. Al aumentar el tamaño del paquete cuando los registros son muy grandes (por ejemplo, al insertar filas en tablas grandes) se aumenta el rendimiento notablemente. El tamaño en bytes es un valor numérico.	
PASSWORD	Especifica si se envían las contraseñas a una fuente de datos. 'Y' Las contraseñas se envían a la fuente de datos y se validan. 'N' Las contraseñas no se envían a la fuente de datos ni se validan.	'Y'

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
PLAN_HINTS	Especifica si se han de habilitar las *indicaciones de planes*. Las indicaciones de planes son fragmentos de sentencias que proporcionan información adicional a los optimizadores de fuentes de datos. Esta información puede, para ciertos tipos de datos, mejorar el rendimiento de consultas. Las indicaciones de planes pueden ayudar al optimizador de fuentes de datos a decidir si se debe utilizar un índice, el índice que se ha de utilizar o qué orden de unión de tablas se ha de utilizar. 'Y' Se han de habilitar las indicaciones de planes en la fuente de datos, si ésta soporta las indicaciones de planes. 'N' No se han de habilitar las indicaciones de planes en la fuente de datos. Esta opción sólo está disponible para las fuentes de datos Oracle y Sybase.	'N'
PORT	Especifica el número del puerto que el reiniciador utiliza para conectarse con el servidor BioRS. Esta opción es opcional.	'5014'
PROCESSORS	Especifica el número de procesadores que el programa HMMER utiliza. Esta opción es equivalente a la opción --cpu del mandato `hmmpfam`.	Ninguno.
PROXY_AUTHID	Especifica el nombre de usuario que debe utilizarse cuando el valor de PROXY_TYPE sea 'SOCKS5'. Este campo es opcional si el valor de PROXY_TYPE es 'SOCKS5'. Póngase en contacto con el administrador de la red para obtener el nombre de usuario que debe utilizarse. Esta opción no es válida si el valor de PROXY_TYPE no es 'SOCKS5'.	Ninguno.
PROXY_PASSWORD	Especifica la contraseña que debe utilizarse cuando el valor de PROXY_TYPE sea 'SOCKS5'. Este campo es opcional si el valor de PROXY_TYPE es 'SOCKS5'. Póngase en contacto con el administrador de la red para obtener la contraseña que debe utilizarse. Esta opción no es válida si el valor de PROXY_TYPE no es 'SOCKS5'.	Ninguno.
PROXY_SERVER_ NAME	Especifica el nombre o la dirección IP del servidor proxy. Esta campo es obligatorio si el valor de PROXY_TYPE es 'HTTP', 'SOCKS4' o 'SOCKS5'. Póngase en contacto con el administrador de la red para obtener el nombre o la dirección IP del servidor proxy.	Ninguno.
PROXY_SERVER_ PORT	Especifica el número de puerto del servidor proxy. Esta campo es obligatorio si el valor de PROXY_TYPE es 'HTTP', 'SOCKS4' o 'SOCKS5'. Póngase en contacto con el administrador de la red para obtener el número de puerto del servidor proxy que debe utilizarse.	Ninguno.

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
PROXY_TYPE	Especifica el tipo de proxy que se utiliza para acceder a Internet cuando se está detrás de un cortafuegos. Los valores válidos son 'NONE', 'HTTP', 'SOCKS4' o 'SOCKS5'. El valor por omisión es 'NONE'. Póngase en contacto con el administrador de la red para obtener el tipo de proxy que se utiliza.	'NONE'
PUSHDOWN	'Y' DB2 UDB tendrá en cuenta la posibilidad de dejar que la fuente de datos evalúe las operaciones. 'N' DB2 UDB enviará las sentencias SQL de la fuente de datos que incluyan sólo SELECT con los nombres de columna. Los predicados (como, por ejemplo, WHERE=) las funciones de columna y escalares (como, por ejemplo, MAX y MIN), las clasificaciones (como, por ejemplo, ORDER BY o GROUP BY) y las uniones no se incluirán en ningún SQL que se envíe a la fuente de datos.	'Y'
RDBMS_TYPE	Especifica el RDBMS que Docbase utiliza. Los valores válidos son DB2, INFORMIX, ORACLE, SQLSERVER o SYBASE. Esta opción es obligatoria.	Ninguno.
RESPONSE_QUEUE	El nombre de la cola de respuesta que entrega los resultados de la consulta del adaptador al reiniciador. El nombre debe cumplir las especificaciones de los nombres de cola para WebSphere MQ. Esta opción es obligatoria.	Ninguno.
REQUEST_QUEUE	El nombre de la cola de peticiones que entrega las peticiones de la consulta del reiniciador al adaptador. El nombre debe cumplir las especificaciones de los nombres de cola para WebSphere MQ. Esta opción es obligatoria.	Ninguno.
SOCKET_TIMEOUT	Especifica el tiempo máximo en minutos que el servidor federado de DB2 esperará los resultados del servidor proxy. Un valor válido es cualquier número mayor o igual a cero. El valor por omisión es cero'0'. Un valor de cero denota un período de tiempo de espera ilimitado.	0
TIMEFORMAT	El formato de hora que la fuente de datos utiliza. El formato debe introducirse utilizando 'hh12', 'hh24', 'mm', 'ss', 'AM' o 'A.M'. Por ejemplo, para representar el formato de hora de '16:00:00', debe utilizarse 'hh24:mm:ss'. Para representar el formato de hora de'8:00:00 AM', debe utilizarse 'hh12:mm:ss AM'. Este campo puede contener nulos.	Ninguno.

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
TIMESTAMPFORMAT	El formato de indicación de fecha y hora que la fuente de datos utiliza. El formato es igual al de la fecha y la hora, más 'n' para las décimas de segundo, 'nn' para las centésimas de segundo, 'nnn' para las milésimas de segundo y así sucesivamente hasta 'nnnnnn' para los microsegundos. Por ejemplo, para representar el formato de indicación de fecha y hora de'2003-01-01-24:00:00.000000', debe utilizarse 'YYYY-MM-DD-hh24:mm:ss.nnnnnn'. Este campo puede contener nulos.	Ninguno.
TIMEOUT	Sybase: Especifica el número de segundos que el servidor federado de DB2 debe esperar una respuesta de Sybase Open Client para cualquier sentencia de SQL. El valor de *segundos* es un número entero positivo dentro del rango de enteros de DB2 Universal Database. El valor de tiempo de espera que debe especificarse depende del reiniciador que se utilice. El comportamiento por omisión de la opción TIMEOUT para los reiniciadores Sybase es 0, que hace que DB2 UDB espere una respuesta de forma indefinida. BioRS: Especifica el tiempo, en minutos, que el reiniciador de BioRS debe esperar una respuesta del servidor BioRS. El valor por omisión es 10. Esta opción es opcional.	'0'; BioRS: '10'
TRANSACTIONS	Especifica la modalidad para las transacciones del servidor. Los valores válidos son: 'NONE' No se permite ninguna transacción. 'QUERY' Sólo se permiten las transacciones para los métodos Dctm_Query. **'ALL'** Se permiten las transacciones para el método Dctm_Query. En este release, ALL tiene la misma función que QUERY.	'QUERY'
USE_CLOB_ SEQUENCE	Esta opción especifica el tipo de datos que el servidor federado utiliza para la columna BlastSeq o HmmQSeq. Los valores pueden ser 'Y' o 'N'. Puede utilizarse la sentencia CREATE NICKNAME o ALTER NICKNAME para alterar temporalmente el tipo de datos por omisión de la columna BlastSeq o HmmQSeq.	'Y'

Tabla 134. Opciones de servidor y los valores de las mismas (continuación)

Opción	Descripción y valores válidos	Valor por omisión
VARCHAR_NO_ TRAILING_BLANKS	Esta opción se aplica a las fuentes de datos que tienen tipos de datos de caracteres variables que no rellenan la longitud con blancos de cola durante la comparación. Algunas fuentes de datos como, por ejemplo, Oracle, carecen de una semántica de comparación con caracteres de relleno de blancos que devuelvan el mismo resultado que la semántica de comparación de DB2 para Linux, UNIX y Windows. Defina esta opción cuando desee que se aplique a todas las columnas VARCHAR y VARCHAR2 de los objetos de la fuente de datos a los que se accederá desde el servidor designado. Esto incluye las vistas. Y — No aparecen blancos de cola en estas columnas VARCHAR o la fuente de datos presenta una semántica de comparación de caracteres de relleno de blancos similar a la semántica del servidor federado. El servidor federado baja las operaciones de comparación de caracteres a la fuente de datos para su proceso N — Aparecen blancos de cola en estas columnas VARCHAR y la fuente de datos presenta una semántica de comparación de caracteres de relleno de blancos diferente de la del servidor federado. El servidor federado procesa las operaciones de comparación de caracteres si no es posible realizar una compensación con una semántica equivalente. Por ejemplo, vuelve a escribir el predicado.	N para las fuentes de datos afectadas.

Notas sobre esta tabla:

1. Este campo se aplica sin tener en cuenta el valor especificado para autentificación.
2. Puesto que DB2 UDB almacena los ID de usuario en mayúsculas, los valores 'N' y 'U' son lógicamente equivalentes.
3. El valor de FOLD_PW no tiene ningún efecto cuando el valor de la contraseña es 'N'. Puesto que no se envía ninguna contraseña, las mayúsculas y minúsculas no pueden ser un factor.
4. Evite los valores nulos para estas opciones. Un valor nulo puede parecer interesante porque DB2 UDB intentará resolver varias veces los ID de usuario y contraseñas; sin embargo, puede afectar negativamente al rendimiento (es posible que DB2 UDB envíe un ID de usuario y contraseña cuatro veces antes de pasar la autentificación de la fuente de datos satisfactoriamente).

Conceptos relacionados:

- "Server characteristics affecting pushdown opportunities" en la publicación *Federated Systems Guide*
- "Server characteristics affecting global optimization" en la publicación *Federated Systems Guide*

Información relacionada:

- "Sentencia DROP" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia ALTER SERVER" en la publicación *Consulta de SQL, Volumen 2*
- "Sentencia CREATE SERVER" en la publicación *Consulta de SQL, Volumen 2*

Opciones de correlación de usuarios para sistemas federados

Estas opciones son válidas para todas las fuentes de datos relacionales. Para las fuentes de datos no relacionales, las opciones REMOTE_AUTHID y REMOTE_PASSWORD son válidas para las fuentes de datos siguientes: BioRS, Documentum, Extended Search y Servicios Web. La opción GUEST es válida para la fuente de datos BioRS.

Estas opciones se utilizan con las sentencias CREATE USER MAPPING y ALTER USER MAPPING.

Tabla 135. Opciones de correlación de usuarios y los valores de las mismas

Opción	Valores válidos	Valor por omisión
ACCOUNTING	DRDA: Se utiliza para especificar una serie de contabilidad DRDA. Los valores válidos son cualquier serie de 255 de longitud o inferior. Esta opción sólo es necesaria si se necesita pasar información de contabilidad. Consulte el manual DB2 Connect Users Guide para obtener más información.	Ninguno
GUEST	Especifica si el reiniciador debe utilizar la modalidad de acceso de huéspedes para el servidor BioRS. **Y** El reiniciador utiliza la modalidad de acceso de huéspedes al servidor BioRS. **N** El reiniciador no utiliza la modalidad de acceso de huéspedes al servidor BioRS. Cuando se establece en el valor Y, esta opción es mutuamente exclusiva con la opción REMOTE_AUTHID y la opción REMOTE_PASSWORD.	N
REMOTE_AUTHID	Indica el ID de autorización utilizado en la fuente de datos. Los valores válidos son cualquier serie de 255 de longitud o inferior.	El ID de autorización que se utilice para la conexión con DB2 Universal Database.
REMOTE_DOMAIN	Documentum: Indica el dominio de Windows NT que se utiliza para autentificar a los usuarios que se conectan a una fuente de datos Documentum. Puede especificarse cualquier nombre de dominio válido de Windows NT.	El dominio de autentificación por omisión para la base de datos de Documentum.

Tabla 135. Opciones de correlación de usuarios y los valores de las mismas (continuación)

Opción	Valores válidos	Valor por omisión
REMOTE_PASSWORD	Indica la contraseña de autorización utilizada en la fuente de datos. Puede especificarse cualquier serie de 32 caracteres de longitud o menos. No es necesario establecer esta opción si se cumplen las condiciones siguientes: • El parámetro de configuración AUTHENTICATON del gestor de bases de datos está definido como SERVER. • Al conectarse a la base de datos DB2, se ha especificado un ID de autentificación y una contraseña. Si el servidor necesita una contraseña y no se define esta opción, es necesario asegurarse de que se cumplan las dos condiciones anteriores; de lo contrario, la conexión fallará.	La contraseña que se utilice para la conexión con DB2 Universal Database si se cumplen las dos condiciones que se muestran en la columna de valores válidos.

Conceptos relacionados:

- "DB2 Connect and DRDA" en la publicación *DB2 Connect User's Guide*
- "DRDA and data access" en la publicación *DB2 Connect User's Guide*

Opciones del reiniciador para sistemas federados

Las opciones del reiniciador se utilizan para configurar el reiniciador o definir la forma en que el servidor federado utiliza el reiniciador. Las opciones del reiniciador pueden establecerse al crear o modificar el reiniciador.

Todas las fuentes de datos relacionales y no relacionales utilizan la opción del reiniciador DB2_FENCED. La fuente de datos ODBC utiliza la opción del reiniciador MODULE. La fuente de datos Entrez utiliza la opción del reiniciador EMAIL.

Tabla 136. Opciones de reiniciador y los valores de las mismas

Opción	Valores válidos		Valor por omisión
DB2_FENCED	Especifica si el reiniciador se ejecuta en modalidad protegida o en modalidad fiable.		Reiniciadores relacionales: N. Reiniciadores no relacionales de IBM: N. Reiniciadores no relacionales de otros fabricantes: Y.
	Y	El reiniciador se ejecuta en modalidad protegida.	
	N	El reiniciador se ejecuta en modalidad fiable.	

Tabla 136. Opciones de reiniciador y los valores de las mismas (continuación)

Opción	Valores válidos	Valor por omisión
EMAIL	Especifica una dirección de correo electrónico al registrar el reiniciador Entrez. Esta dirección de correo electrónico se incluye con todas las consultas y permite a NCBI contactar con el usuario si surgen problemas como, por ejemplo, si demasiadas consultas sobrecargan los servidores de NCBI. Esta opción es obligatoria.	
MODULE	Especifica la vía de acceso completa de la biblioteca que contiene la implementación del Gestor de controladores ODBC o la implementación de SQL/CLI. Se necesita para el reiniciador de ODBC en los servidores federados de UNIX.	En Windows, el valor por omisión es odbc32.dll

Tareas relacionadas:

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

Correlaciones de tipos de datos en avance por omisión

Los dos tipos de correlaciones entre los tipos de datos de la fuente de datos y los tipos de datos de la base de datos federada son las correlaciones de tipos en avance y las correlaciones de tipos invertidas. En una *correlación de tipos en avance*, la correlación se realiza de un tipo remoto a un tipo local comparable.

Es posible alterar temporalmente una correlación de tipos por omisión o crear una correlación de tipos nueva con la sentencia CREATE TYPE MAPPING.

Estas correlaciones son válidas con todas las versiones a las que se da soporte, a menos que se indique lo contrario.

Para todas las correlaciones de tipos de datos en avance por omisión de una fuente de datos a DB2 para Linux, UNIX y Windows, el esquema federado de DB2 es SYSIBM.

Las tablas siguientes muestran las correlaciones en avance por omisión entre los tipos de datos DB2 para Linux, UNIX y Windows y los tipos de datos de la fuente de datos.

Fuentes de datos DB2 para z/OS y OS/390

Tabla 137. Correlaciones de tipos de datos en avance por omisión de DB2 para z/OS y OS/390 (No se muestran todas las columnas)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
ROWID	-	-	-	-	Y	-	VARCHAR	40	-	Y
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

Fuentes de datos DB2 para iSeries

Tabla 138. Correlaciones de tipos de datos en avance por omisión de DB2 para iSeries (No se muestran todas las columnas)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
NUMERIC	-	-	-	-	-	-	DECIMAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

Fuentes de datos DB2 Server para VM y VSE

Tabla 139. Correlaciones de tipos de datos en avance por omisión de DB2 Server para VM y VSE (No se muestran todas las columnas)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBAHW	-	-	-	-	-	-	SMALLINT	-	0	-
DBAINT	-	-	-	-	-	-	INTEGER	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPH	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

Fuentes de datos DB2 para Linux, UNIX y Windows

Tabla 140. Correlaciones de tipos de datos en avance por omisión de DB2 para Linux, UNIX y Windows (No se muestran todas las columnas)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BIGINT	-	-	-	-	-	-	BIGINT	-	0	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	-	-	-	-	-	-	CHAR	-	0	N
CHAR	-	-	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
LONGVAR	-	-	-	-	N	-	CLOB	-	-	-
LONGVAR	-	-	-	-	Y	-	BLOB	-	-	-
LONGVARG	-	-	-	-	-	-	DBCLOB	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	0	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	0	N

Fuentes de datos Informix

Tabla 141. Correlaciones de tipos de datos en avance por omisión de Informix (No se muestran todas las columnas)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	2147483647	-	-
BOOLEAN	-	-	-	-	-	-	CHARACTER	1	-	-
BYTE	-	-	-	-	-	-	BLOB	2147483647	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CLOB	-	-	-	-	-	-	CLOB	2147483647	-	-
DATE	-	-	-	-	-	-	DATE	4	-	-
DATETIME	0	4	0	4	-	-	DATE	4	-	-
DATETIME	6	10	6	10	-	-	TIME	3	-	-
DATETIME	0	4	6	15	-	-	TIMESTAMP	10	-	-
DATETIME	6	10	11	15	-	-	TIMESTAMP	10	-	-
DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
DECIMAL	32	130	-	-	-	-	DOUBLE	8	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
INTERVAL	-	-	-	-	-	-	VARCHAR	25	-	-
INT8	-	-	-	-	-	-	BIGINT	19	0	-
LVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
MONEY	1	31	0	31	-	-	DECIMAL	-	-	-
MONEY	32	32	-	-	-	-	DOUBLE	8	-	-
NCHAR	1	254	-	-	-	-	CHARACTER	-	-	-
NCHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
NVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
REAL	-	-	-	-	-	-	REAL	4	-	-
SERIAL	-	-	-	-	-	-	INTEGER	4	-	-
SERIAL8	-	-	-	-	-	-	BIGINT	-	-	-
SMALLFLOAT	-	-	-	-	-	-	REAL	4	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
TEXT	-	-	-	-	-	-	CLOB	2147483647	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-

Tabla 141. Correlaciones de tipos de datos en avance por omisión de Informix (No se muestran todas las columnas) (continuación)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA

Notas:

- Para el tipo de datos DATETIME de Informix, el servidor federado DB2 para UNIX y Windows usa el calificador de alto nivel de Informix como REMOTE_LENGTH y el calificador de bajo nivel de Informix como REMOTE_SCALE.

  Los calificadores de Informix son las constantes "TU_" definidas en el archivo `datatime.h` del SDK del cliente Informix. Las constantes son:

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

Fuentes de datos Microsoft SQL Server

Tabla 142. Correlaciones de tipos de datos en avance por omisión de Microsoft SQL Server

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
bigint [4]	-	-	-	-	-	-	BIGINT	-	-	-
binary	1	254	-	-	-	-	CHARACTER	-	-	Y
binary	255	8000	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	2	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	8000	-	-	-	-	VARCHAR	-	-	N
fecha y hora	-	-	-	-	-	-	TIMESTAMP	10	-	-
datetimen	-	-	-	-	-	-	TIMESTAMP	10	-	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-

Tabla 142. Correlaciones de tipos de datos en avance por omisión de Microsoft SQL Server (continuación)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
DUMMY65 [1]	1	38	-84	127	-	-	DOUBLE	-	-	-
DUMMY2000 [3]	1	38	-84	127	-	-	DOUBLE	-	-	-
float	-	8	-	-	-	-	DOUBLE	8	-	-
floatn	-	8	-	-	-	-	DOUBLE	8	-	-
float	-	4	-	-	-	-	REAL	4	-	-
floatn	-	4	-	-	-	-	REAL	4	-	-
image	-	-	-	-	-	-	BLOB	2147483647	-	Y
int	-	-	-	-	-	-	INTEGER	4	-	-
intn	-	-	-	-	-	-	INTEGER	4	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	127	-	-	-	-	CHAR	-	-	N
nchar	128	4000	-	-	-	-	VARCHAR	-	-	N
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	8	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
ntext [2]	-	-	-	-	-	-	CLOB	2147483647	-	Y
nvarchar	1	4000	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	4	-	-
smallint	-	-	-	-	-	-	SMALLINT	2	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP	10	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
smallmoneyn	-	-	-	-	-	-	DECIMAL	10	4	-
SQL_BIGINT	-	-	-	-	-	-	DECIMAL	-	-	-
SQL_BIGINT [4]	-	-	-	-	-	-	BIGINT	-	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N

Tabla 142. Correlaciones de tipos de datos en avance por omisión de Microsoft SQL Server (continuación)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
SQL_CHAR	255	8000	-	-	-	-	VARCHAR	-	-	N
SQL_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DECIMAL	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_GUID [2]	1	4000	-	-	Y	-	VARCHAR	16	-	Y
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_REAL	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	8000	-	-	-	-	VARCHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	N
timestamp	-	-	-	-	-	-	VARCHAR	8		Y
tinyint	-	-	-	-	-	-	SMALLINT	2	-	-
uniqueidentifier [2]	1	4000	-	-	Y	-	VARCHAR	16	-	Y
varbinary	1	8000	-	-	-	-	VARCHAR	-	-	Y
varchar	1	8000	-	-	-	-	VARCHAR	-	-	N

Notas:

1. Esta correlación de tipos sólo es válida con Microsoft SQL Server Versión 6.5.
2. Esta correlación de tipos sólo es válida con Microsoft SQL Server Versión 7 y Versión 2000.
3. Esta correlación de tipos sólo es válida con los sistemas operativos Windows 2000.
4. Esta correlación de tipos sólo es válida con Microsoft SQL Server Versión 2000.

Fuentes de datos ODBC

Tabla 143. Correlaciones de tipos de datos en avance por omisión de ODBC (No se muestran todas las columnas)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
SQL_BIGINT	-	-	-	-	-	-	BIGINT	8	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	32672	-	-	-	-	VARCHAR	-	-	N
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	4	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TYPE_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_TYPE_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TYPE_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	127	-	-	-	-	CHAR	-	-	N
SQL_WCHAR	128	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WLONGVARCHAR	-	1073741823	-	-	-	-	CLOB	2147483647	-	N

Fuentes de datos NET8 de Oracle

Tabla 144. Correlaciones de tipos de datos en avance por omisión de Oracle NET8

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BLOB	0	0	0	0	-	\0	BLOB	2147483647	0	Y
CHAR	1	254	0	0	-	\0	CHAR	0	0	N
CHAR	255	2000	0	0	-	\0	VARCHAR	0	0	N
CLOB	0	0	0	0	-	\0	CLOB	2147483647	0	N
DATE	0	0	0	0	-	\0	TIMESTAMP	0	0	N
FLOAT	1	126	0	0	-	\0	DOUBLE	0	0	N
LONG	0	0	0	0	-	\0	CLOB	2147483647	0	N
LONG RAW	0	0	0	0	-	\0	BLOB	2147483647	0	Y
MLSLABEL	0	0	0	0	-	\0	VARCHAR	255	0	N
NUMBER	1	38	-84	127	-	\0	DOUBLE	0	0	N
NUMBER	1	31	0	31	-	>=	DECIMAL	0	0	N
NUMBER	1	4	0	0	-	\0	SMALLINT	0	0	N
NUMBER	5	9	0	0	-	\0	INTEGER	0	0	N
NUMBER	-	10	0	0	-	\0	DECIMAL	0	0	N
RAW	1	2000	0	0	-	\0	VARCHAR	0	0	Y
ROWID	0	0	0	NULL	-	\0	CHAR	18	0	N
TIMESTAMP [1]	-	-	-	-	-	-	TIMESTAMP	10	-	-
VARCHAR2	1	4000	0	0	-	\0	VARCHAR	0	0	N

Notas:

1. Esta correlación de tipos sólo es válida para las configuraciones de cliente y servidor de Oracle 9i (o posterior).

Fuentes de datos Sybase

Tabla 145. Correlaciones de tipos de datos en avance por omisión de CTLIB de Sybase

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
binary	1	254	-	-	-	-	CHAR	-	-	Y
binary	255	16384	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	-	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	16384	-	-	-	-	VARCHAR	-	-	N
charx nulo (vea varchar)										
fecha y hora	-	-	-	-	-	-	TIMESTAMP	-	-	-
datetimn	-	-	-	-	-	-	TIMESTAMP	-	-	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
float	-	4	-	-	-	-	REAL	-	-	-
float	-	8	-	-	-	-	DOUBLE	-	-	-
floatn	-	4	-	-	-	-	REAL	-	-	-
floatn	-	8	-	-	-	-	DOUBLE	-	-	-
image	-	-	-	-	-	-	BLOB	-	-	-
int	-	-	-	-	-	-	INTEGER	-	-	-
intn	-	-	-	-	-	-	INTEGER	-	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	254	-	-	-	-	CHAR	-	-	N
nchar	255	16384	-	-	-	-	VARCHAR	-	-	N
nchar nulo (vea nvarchar)										
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
nvarchar	1	16384	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	-	-	-

Tabla 145. Correlaciones de tipos de datos en avance por omisión de CTLIB de Sybase (continuación)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
smalldatetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
smallint	-	-	-	-	-	-	SMALLINT	-	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
sysname	1	254	-	-	-	-	CHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	-
timestamp	-	-	-	-	-	-	VARCHAR	8	-	Y
tinyint	-	-	-	-	-	-	SMALLINT	-	-	-
unichar[1]	1	254	-	-	-	-	CHAR	-	-	N
unichar[1]	255	16384	-	-	-	-	VARCHAR	-	-	N
unichar nulo (vea univarchar)										
univarchar[1]	1	16384	-	-	-	-	VARCHAR	-	-	N
varbinary	1	16384	-	-	-	-	VARCHAR	-	-	Y
varchar	1	16384	-	-	-	-	VARCHAR	-	-	N

Notas:

1. Válido para bases de datos federadas que no son Unicode.

Fuentes de datos Teradata

Tabla 146. Correlaciones de tipos de datos en avance por omisión de Teradata (No se muestran todas las columnas)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BYTE	1	254	-	-	-	-	CHAR	-	-	Y
BYTE	255	32672	-	-	-	-	VARCHAR	-	-	Y
BYTE	32673	64000	-	-	-	-	BLOB	-	-	-
BYTEINT	-	-	-	-	-	-	SMALLINT	-	-	-

Tabla 146. Correlaciones de tipos de datos en avance por omisión de Teradata (No se muestran todas las columnas) (continuación)

REMOTE_TYPENAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPENAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CHAR	32673	64000	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DOUBLE PRECISION	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	-	-
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	-	-
GRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
INTERVAL	-	-	-	-	-	-	CHAR	-	-	-
NUMERIC	1	18	0	18	-	-	DECIMAL	-	-	-
REAL	-	-	-	-	-	-	DOUBLE	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	-	-
VARBYTE	1	32762	-	-	-	-	VARCHAR	-	-	Y
VARBYTE	32763	64000	-	-	-	-	BLOB	-	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
VARCHAR	32673	64000	-	-	-	-	CLOB	-	-	-
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	-	-
VARGRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-

Conceptos relacionados:

- "Forward and reverse data type mappings" en la publicación *Federated Systems Guide*

Información relacionada:

- "Altering long data types to varchar data types" en la publicación *Federated Systems Guide*
- "Unicode default forward data type mappings - NET8 wrapper" en la publicación *Federated Systems Guide*

- "Unicode default forward data type mappings - Sybase wrapper" en la publicación *Federated Systems Guide*
- "Unicode default forward data type mappings - ODBC wrapper" en la publicación *Federated Systems Guide*
- "Unicode default forward data type mappings - Microsoft SQL Server wrapper" en la publicación *Federated Systems Guide*

Correlaciones de tipos de datos invertidas por omisión

Los dos tipos de correlaciones entre los tipos de datos de la fuente de datos y los tipos de datos de la base de datos federada son las correlaciones de tipos en avance y las correlaciones de tipos invertidas. En una *correlación de tipos en avance*, la correlación se realiza de un tipo remoto a un tipo local comparable. El otro tipo de correlación es una *correlación de tipos invertida*, que se utiliza con un DDL transparente para crear o modificar tablas remotas.

Para la mayor parte de fuentes de datos, las correlaciones de tipos por omisión se encuentran en los reiniciadores. Las correlaciones de tipo por omisión para las fuentes de datos de la familia DB2 se encuentran en el reiniciador DRDA. Las correlaciones de tipos por omisión para Informix se encuentran en el reiniciador INFORMIX y así sucesivamente.

Cuando se define una tabla o una vista remota en la base de datos federada de DB2, la definición incluye una correlación de tipos invertida. La correlación es entre el tipo de datos *de* DB2 para Linux, UNIX y Windows local para cada columna y el tipo de datos *remoto* correspondiente. Por ejemplo, existe una correlación de tipos invertida por omisión en la que el tipo local REAL hace referencia al tipo SMALLFLOAT de Informix.

Los servidores federados de DB2 para Linux, UNIX y Windows no proporcionan soporte a las correlaciones para LONG VARCHAR, LONG VARGRAPHIC, DATALINK ni para los tipos definidos por el usuario.

Cuando se utiliza la sentencia CREATE TABLE para crear una tabla remota, deben especificarse los tipos de datos locales que se desea incluir en la tabla remota. Estas correlaciones de tipos invertidas por omisión asignación los tipos remotos correspondientes a estas columnas. Por ejemplo, supongamos que se utiliza la sentencia CREATE TABLE para definir una tabla Informix con una columna C2. Se especifica BIGINT como tipo de datos para C2 en la sentencia. La correlación de tipos invertida por omisión de BIGINT depende de la versión de Informix en la que esté creándose la tabla. La correlación para C2 en la tabla Informix será DECIMAL en Informix Versión 8 e INT8 en Informix Versión 9.

Es posible alterar temporalmente una correlación de tipos invertida por omisión o crear una correlación de tipos invertida nueva con la sentencia CREATE TYPE MAPPING.

Las tablas siguientes muestran las correlaciones invertidas por omisión entre los tipos de datos locales DB2 para Linux, UNIX y Windows y los tipos de datos de fuentes de datos remotas.

Estas correlaciones son válidas con todas las versiones a las que se da soporte, a menos que se indique lo contrario.

Fuentes de datos DB2 para z/OS y OS/390

Tabla 147. Correlaciones de tipos de datos invertidas por omisión de DB2 para z/OS y OS/390 (No se muestran todas las columnas)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	–
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	N

Fuentes de datos DB2 para iSeries

Tabla 148. Correlaciones de tipos de datos invertidas por omisión de DB2 para iSeries (No se muestran todas las columnas)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	N
CHARACTER	-	-	-	-	Y	-	CHARACTER	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	NUMERIC	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARG	-	-	N

Fuentes de datos DB2 para VM y VSE

Tabla 149. Correlaciones de tipos de datos invertidas por omisión de DB2 para VM y VSE (No se muestran todas las columnas)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPH	-	-	N

Fuentes de datos DB2 para Linux, UNIX y Windows

Tabla 150. Correlaciones de tipos de datos invertidas por omisión de DB2 para Linux, UNIX y Windows (No se muestran todas las columnas)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	FEDERATED_BIT_DATA
BIGINT	-	8	-	-	-	-	BIGINT	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	-	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

Fuentes de datos Informix

Tabla 151. Correlaciones de tipos de datos invertidas por omisión de Informix

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BIGINT [1]	-	-	-	-	-	-	DECIMAL	19	-	-
BIGINT [2]	-	-	-	-	-	-	INT8	-	-	-
BLOB	1	2147483647	-	-	-	-	BYTE	-	-	-
CHARACTER	-	-	-	-	N	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB	1	2147483647	-	-	-	-	TEXT	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	SMALLFLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	DATETIME	6	10	-
TIMESTAMP	-	10	-	-	-	-	DATETIME	0	15	-
VARCHAR	1	254	-	-	N	-	VARCHAR	-	-	-
VARCHAR	255	32672	-	-	N	-	TEXT	-	-	-
VARCHAR	-	-	-	-	Y	-	BYTE	-	-	-
VARCHAR [2]	255	2048	-	-	N	-	LVARCHAR	-	-	-
VARCHAR [2]	2049	32672	-	-	N	-	TEXT	-	-	-

Notas:

1. Esta correlación de tipos sólo es válida con el servidor Informix Versión 8 (o inferior).
2. Esta correlación de tipos sólo es válida con el servidor Informix Versión 9.

Para el tipo de datos DATETIME de Informix, el servidor federado DB2 para UNIX y Windows usa el calificador de alto nivel de Informix como REMOTE_LENGTH y el calificador de bajo nivel de Informix como REMOTE_SCALE.

Los calificadores de Informix son las constantes "TU_" definidas en el archivo `datatime.h` del SDK del cliente Informix. Las constantes son:

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

Fuentes de datos de Microsoft SQL Server

Tabla 152. Correlaciones de tipos de datos invertidas por omisión de Microsoft SQL Server (No se muestran todas las columnas)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BIGINT [1]	-	-	-	-	-	-	bigint	-	-	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	4	-	-	-	-	fecha y hora	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	8	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	int	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
REAL	-	4	-	-	-	-	real	-	-	-
TIME	-	3	-	-	-	-	fecha y hora	-	-	-
TIMESTAMP	-	10	-	-	-	-	fecha y hora	-	-	-
VARCHAR	1	8000	-	-	N	-	varchar	-	-	-
VARCHAR	8001	32672	-	-	N	-	text	-	-	-
VARCHAR	1	8000	-	-	Y	-	varbinary	-	-	-
VARCHAR	8001	32672	-	-	Y	-	image	-	-	-

Notas:

1. Esta correlación de tipos sólo es válida con Microsoft SQL Server Versión 2000.

Fuentes de datos NET8 de Oracle

Tabla 153. Correlaciones de tipos de datos invertidas por omisión de NET8 de Oracle

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BLOB	0	2147483647	0	0	Y	\0	BLOB	0	0	Y
CHARACTER	1	254	0	0	N	\0	CHAR	0	0	N
CHARACTER	1	254	0	0	Y	\0	RAW	0	0	Y
CLOB	0	2147483647	0	0	N	\0	CLOB	0	0	N
DATE	0	4	0	0	N	\0	DATE	0	0	N
DECIMAL	0	0	0	0	N	\0	NUMBER	0	0	N
DOUBLE	0	8	0	0	N	\0	FLOAT	126	0	N
FLOAT	0	8	0	0	N	\0	FLOAT	126	0	N
INTEGER	0	4	0	0	N	\0	NUMBER	9	0	N
REAL	0	4	0	0	N	\0	FLOAT	63	0	N
SMALLINT	0	2	0	0	N	\0	NUMBER	4	0	N
TIME	0	3	0	0	N	\0	DATE	0	0	N
TIMESTAMP	0	10	0	0	N	\0	DATE	0	0	N
VARCHAR	1	4000	0	0	N	\0	VARCHAR2	0	0	N
VARCHAR	1	2000	0	0	Y	\0	RAW	0	0	Y

Nota: El tipo de datos BIGINT de DB2 para Linux, UNIX y Windows no está disponible para un DDL transparente. No es posible especificar el tipo de datos BIGINT en una sentencia CREATE TABLE al crear una tabla Oracle remota.

Fuentes de datos Sybase

Tabla 154. Correlaciones de tipos de datos invertidas por omisión de CTLIB de Sybase

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	REMOTE_BIT_DATA
BIGINT	-	-	-	-	-	-	decimal	19	0	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	-	-	-	-	-	fecha y hora	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	-	-	-	-	-	float	-	-	-
GRAPHIC	-	-	-	-	-	-	unichar	-	-	-
VARGRAPHIC	-	-	-	-	-	-	univarchar	-	-	-
INTEGER	-	-	-	-	-	-	entero	-	-	-
REAL	-	-	-	-	-	-	real	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
TIME	-	-	-	-	-	-	fecha y hora	-	-	-
TIMESTAMP	-	-	-	-	-	-	fecha y hora	-	-	-
VARCHAR[1]	1	255	-	-	N	-	varchar	-	-	-
VARCHAR[1]	256	32672	-	-	N	-	text	-	-	-
VARCHAR [2]	1	16384	-	-	N	-	varchar	-	-	-
VARCHAR [2]	16385	32672	-	-	N	-	text	-	-	-
VARCHAR[1]	1	255	-	-	Y	-	varbinary	-	-	-
VARCHAR[1]	256	32672	-	-	Y	-	image	-	-	-
VARCHAR [2]	1	16384	-	-	Y	-	varbinary	-	-	-
VARCHAR [2]	16385	32672	-	-	Y	-	image	-	-	-

Notas:

1. Esta correlación de tipos sólo es válida para CTLIB con el servidor de Sybase versión 12.0 (o anterior).
2. Esta correlación de tipos sólo es válida para CTLIB con el servidor de Sybase versión 12.5 (o posterior).

Fuentes de datos Teradata

Tabla 155. Correlaciones de tipos de datos invertidas por omisión de Teradata (No se muestran todas las columnas)

FEDERATED_TYPENAME	FEDERATED_LOWER_LEN	FEDERATED_UPPER_LEN	FEDERATED_LOWER_SCALE	FEDERATED_UPPER_SCALE	FEDERATED_BIT_DATA	FEDERATED_DATA_OPERATORS	REMOTE_TYPENAME	REMOTE_LENGTH	REMOTE_SCALE	FEDERATED_BIT_DATA
BLOB [1]	1	64000	-	-	-	-	VARBYTE	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB [2]	1	64000	-	-		-	VARCHAR	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DBCLOB [3]	1	32000	-	-	-	-	VARGRAPHIC	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DECIMAL	19	31	0	31	-	-	FLOAT	-	-	-
DOUBLE	-	-	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARBYTE	-	-	-
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

Notas:

1. El tipo de datos VARBYTE de Teradata sólo puede contener la longitud especificada (entre 1 y 64000) de un tipo de datos BLOB de DB2.
2. El tipo de datos VARCHAR de Teradata sólo puede contener la longitud especificada (entre 1 y 64000) de un tipo de datos CLOB de DB2.
3. El tipo de datos VARGRAPHIC de Teradata sólo puede contener la longitud especificada (entre 1 y 32000) de un tipo de datos DBCLOB de DB2.

Conceptos relacionados:

- "Forward and reverse data type mappings" en la publicación *Federated Systems Guide*

Apéndice F. La base de datos SAMPLE

Muchos de los ejemplos de códigos en la documentación de DB2 utilizan la base de datos SAMPLE. A continuación se proporciona una descripción de cada una de las tablas de la base de datos SAMPLE. También se proporcionan instrucciones para crear y soltar la base de datos. Se dan los valores iniciales de los datos para cada tabla; un guión (–) indica un valor nulo (NULL).

Creación de la base de datos SAMPLE

Utilice el mandato DB2SAMPL para crear la base de datos SAMPLE. Para crear una base de datos, debe tener autorización SYSADM.

- **Cuando se utilizan plataformas basadas en UNIX**

  Si utiliza el indicador de mandatos del sistema operativo, escriba:

  ```
  sqllib/bin/db2sampl <vía>
  ```

  en el directorio inicial del propietario de la instancia del gestor de bases de datos, donde *vía de acceso* es un parámetro opcional que especifica la vía de acceso donde se ha de crear la base de datos SAMPLE. Si no se especifica el parámetro de vía de acceso, la base de datos de muestra se crea en la vía de acceso por omisión especificada por el parámetro DFTDBPATH en el archivo de configuración del gestor de bases de datos. El esquema para DB2SAMPL es el valor del registro especial CURRENT SCHEMA.
- **Cuando se utilizan plataformas de Windows**

  Si utiliza el indicador de mandatos del sistema operativo, escriba:

  ```
  db2sampl e
  ```

  donde *e* es un parámetro opcional que especifica la unidad donde se ha de crear la base de datos. Si no se especifica el parámetro de unidad, la base de datos de muestra se crea en la misma unidad que DB2.

Eliminación de la base de datos SAMPLE

Si no necesita acceder a la base de datos SAMPLE, puede borrarla utilizando el mandato DROP DATABASE:

```
db2 drop database sample
```

Tabla CL_SCHED

Nombre:	CLASS_CODE	DAY	STARTING	ENDING
Tipo:	char(7)	smallint	time	time
Desc:	Class Code (room:teacher)	Day # of 4 day schedule	Class Start Time	Class End Time

Tabla DEPARTMENT

Nombre:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
Tipo:	char(3) not null	varchar(29) not null	char(6)	char(3) not null	char(16)

Tabla DEPARTMENT

Nombre:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
Desc:	Department number	Name describing general activities of department	Employee number (EMPNO) of department manager	Department (DEPTNO) to which this department reports	Name of the remote location
Valores:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	-
	B01	PLANNING	000020	A00	-
	C01	INFORMATION CENTER	000030	A00	-
	D01	DEVELOPMENT CENTER	-	A00	-
	D11	MANUFACTURING SYSTEMS	000060	D01	-
	D21	ADMINISTRATION SYSTEMS	000070	D01	-
	E01	SUPPORT SERVICES	000050	A00	-
	E11	OPERATIONS	000090	E01	-
	E21	SOFTWARE SUPPORT	000100	E01	-

Tabla EMPLOYEE

Nombres:	EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
Tipo:	char(6) not null	varchar(12) not null	char(1) not null	varchar(15) not null	char(3)	char(4)	date
Desc:	Employee number	First name	Middle initial	Last name	Department (DEPTNO) in which the employee works	Phone number	Date of hire

JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(8)	smallint not null	char(1)	date	dec(9,2)	dec(9,2)	dec(9,2)
Job	Number of years of formal education	Sex (M male, F female)	Date of birth	Yearly salary	Yearly bonus	Yearly commission

La tabla siguiente contiene los valores de la tablaEMPLOYEE.

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(6) not null	varchar(12) not null	char(1) not null	varchar(15) not null	char(3)	char(4)	date	char(8)	smallint not null	char(1)	date	dec(9,2)	dec(9,2)	dec(9,2)
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRES	18	F	1933-08-24	52750	1000	4220
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250	800	3300
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250	800	3060
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175	800	3214
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250	500	2580
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170	700	2893
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750	600	2380
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150	500	2092
000110	VINCENZO	G	LUCCHESSI	A00	3490	1958-05-16	SALESREP	19	M	1929-11-05	46500	900	3720
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLERK	14	M	1942-10-18	29250	600	2340
000130	DOLORES	M	QUINTANA	C01	4578	1971-07-28	ANALYST	16	F	1925-09-15	23800	500	1904
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420	600	2274
000150	BRUCE		ADAMSON	D11	4510	1972-02-12	DESIGNER	16	M	1947-05-17	25280	500	2022
000160	ELIZABETH	R	PIANKA	D11	3782	1977-10-11	DESIGNER	17	F	1955-04-12	22250	400	1780
000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680	500	1974
000180	MARILYN	S	SCOUTTEN	D11	1682	1973-07-07	DESIGNER	17	F	1949-02-21	21340	500	1707
000190	JAMES	H	WALKER	D11	2986	1974-07-26	DESIGNER	16	M	1952-06-25	20450	400	1636
000200	DAVID		BROWN	D11	4501	1966-03-03	DESIGNER	16	M	1941-05-29	27740	600	2217
000210	WILLIAM	T	JONES	D11	0942	1979-04-11	DESIGNER	17	M	1953-02-23	18270	400	1462
000220	JENNIFER	K	LUTZ	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840	600	2387
000230	JAMES	J	JEFFERSON	D21	2094	1966-11-21	CLERK	14	M	1935-05-30	22180	400	1774
000240	SALVATORE	M	MARINO	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760	600	2301
000250	DANIEL	S	SMITH	D21	0961	1969-10-30	CLERK	15	M	1939-11-12	19180	400	1534
000260	SYBIL	P	JOHNSON	D21	8953	1975-09-11	CLERK	16	F	1936-10-05	17250	300	1380
000270	MARIA	L	PEREZ	D21	9001	1980-09-30	CLERK	15	F	1953-05-26	27380	500	2190
000280	ETHEL	R	SCHNEIDER	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250	500	2100
000290	JOHN	R	PARKER	E11	4502	1980-05-30	OPERATOR	12	M	1946-07-09	15340	300	1227
000300	PHILIP	X	SMITH	E11	2095	1972-06-19	OPERATOR	14	M	1936-10-27	17750	400	1420
000310	MAUDE	F	SETRIGHT	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900	300	1272
000320	RAMLAL	V	MEHTA	E21	9990	1965-07-07	FIELDREP	16	M	1932-08-11	19950	400	1596
000330	WING		LEE	E21	2103	1976-02-23	FIELDREP	14	M	1941-07-18	25370	500	2030
000340	JASON	R	GOUNOT	E21	5698	1947-05-05	FIELDREP	16	M	1926-05-17	23840	500	1907

Tabla EMP_ACT

Nombre:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
Tipo:	char(6) not null	char(6) not null	smallint not null	dec(5,2)	date	date
Desc:	Employee number	Project number	Activity number	Proportion of employee's time spent on project	Date activity starts	Date activity ends
Valores:	000010	AD3100	10	.50	1982-01-01	1982-07-01
	000070	AD3110	10	1.00	1982-01-01	1983-02-01
	000230	AD3111	60	1.00	1982-01-01	1982-03-15
	000230	AD3111	60	.50	1982-03-15	1982-04-15
	000230	AD3111	70	.50	1982-03-15	1982-10-15
	000230	AD3111	80	.50	1982-04-15	1982-10-15
	000230	AD3111	180	1.00	1982-10-15	1983-01-01
	000240	AD3111	70	1.00	1982-02-15	1982-09-15
	000240	AD3111	80	1.00	1982-09-15	1983-01-01
	000250	AD3112	60	1.00	1982-01-01	1982-02-01
	000250	AD3112	60	.50	1982-02-01	1982-03-15
	000250	AD3112	60	.50	1982-12-01	1983-01-01
	000250	AD3112	60	1.00	1983-01-01	1983-02-01
	000250	AD3112	70	.50	1982-02-01	1982-03-15
	000250	AD3112	70	1.00	1982-03-15	1982-08-15
	000250	AD3112	70	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.50	1982-10-15	1982-12-01
	000250	AD3112	180	.50	1982-08-15	1983-01-01
	000260	AD3113	70	.50	1982-06-15	1982-07-01
	000260	AD3113	70	1.00	1982-07-01	1983-02-01
	000260	AD3113	80	1.00	1982-01-01	1982-03-01
	000260	AD3113	80	.50	1982-03-01	1982-04-15
	000260	AD3113	180	.50	1982-03-01	1982-04-15
	000260	AD3113	180	1.00	1982-04-15	1982-06-01
	000260	AD3113	180	.50	1982-06-01	1982-07-01
	000270	AD3113	60	.50	1982-03-01	1982-04-01
	000270	AD3113	60	1.00	1982-04-01	1982-09-01
	000270	AD3113	60	.25	1982-09-01	1982-10-15
	000270	AD3113	70	.75	1982-09-01	1982-10-15
	000270	AD3113	70	1.00	1982-10-15	1983-02-01
	000270	AD3113	80	1.00	1982-01-01	1982-03-01
	000270	AD3113	80	.50	1982-03-01	1982-04-01
	000030	IF1000	10	.50	1982-06-01	1983-01-01
	000130	IF1000	90	1.00	1982-01-01	1982-10-01
	000130	IF1000	100	.50	1982-10-01	1983-01-01
	000140	IF1000	90	.50	1982-10-01	1983-01-01
	000030	IF2000	10	.50	1982-01-01	1983-01-01
	000140	IF2000	100	1.00	1982-01-01	1982-03-01
	000140	IF2000	100	.50	1982-03-01	1982-07-01

Nombre:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000140	IF2000	110	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-10-01	1983-01-01
	000010	MA2100	10	.50	1982-01-01	1982-11-01
	000110	MA2100	20	1.00	1982-01-01	1982-03-01
	000010	MA2110	10	1.00	1982-01-01	1983-02-01
	000200	MA2111	50	1.00	1982-01-01	1982-06-15
	000200	MA2111	60	1.00	1982-06-15	1983-02-01
	000220	MA2111	40	1.00	1982-01-01	1983-02-01
	000150	MA2112	60	1.00	1982-01-01	1982-07-15
	000150	MA2112	180	1.00	1982-07-15	1983-02-01
	000170	MA2112	60	1.00	1982-01-01	1983-06-01
	000170	MA2112	70	1.00	1982-06-01	1983-02-01
	000190	MA2112	70	1.00	1982-02-01	1982-10-01
	000190	MA2112	80	1.00	1982-10-01	1983-10-01
	000160	MA2113	60	1.00	1982-07-15	1983-02-01
	000170	MA2113	80	1.00	1982-01-01	1983-02-01
	000180	MA2113	70	1.00	1982-04-01	1982-06-15
	000210	MA2113	80	.50	1982-10-01	1983-02-01
	000210	MA2113	180	.50	1982-10-01	1983-02-01
	000050	OP1000	10	.25	1982-01-01	1983-02-01
	000090	OP1010	10	1.00	1982-01-01	1983-02-01
	000280	OP1010	130	1.00	1982-01-01	1983-02-01
	000290	OP1010	130	1.00	1982-01-01	1983-02-01
	000300	OP1010	130	1.00	1982-01-01	1983-02-01
	000310	OP1010	130	1.00	1982-01-01	1983-02-01
	000050	OP2010	10	.75	1982-01-01	1983-02-01
	000100	OP2010	10	1.00	1982-01-01	1983-02-01
	000320	OP2011	140	.75	1982-01-01	1983-02-01
	000320	OP2011	150	.25	1982-01-01	1983-02-01
	000330	OP2012	140	.25	1982-01-01	1983-02-01
	000330	OP2012	160	.75	1982-01-01	1983-02-01
	000340	OP2013	140	.50	1982-01-01	1983-02-01
	000340	OP2013	170	.50	1982-01-01	1983-02-01
	000020	PL2100	30	1.00	1982-01-01	1982-09-15

Tabla EMP_PHOTO

Nombre:	EMPNO	PHOTO_FORMAT	PICTURE
Tipo:	char(6) not null	varchar(10) not null	blob(100k)
Desc:	Employee number	Photo format	Photo of employee
Valores:	000130	bitmap	db200130.bmp
	000130	gif	db200130.gif
	000130	xwd	db200130.xwd
	000140	bitmap	db200140.bmp
	000140	gif	db200140.gif

Tabla EMP_PHOTO

Nombre:	EMPNO	PHOTO_FORMAT	PICTURE
	000140	xwd	db200140.xwd
	000150	bitmap	db200150.bmp
	000150	gif	db200150.gif
	000150	xwd	db200150.xwd
	000190	bitmap	db200190.bmp
	000190	gif	db200190.gif
	000190	xwd	db200190.xwd

Tabla EMP_RESUME

Nombre:	EMPNO	RESUME_FORMAT	RESUME
Tipo:	char(6) not null	varchar(10) not null	clob(5k)
Desc:	Employee number	Resume Format	Resume of employee
Valores:	000130	ascii	db200130.asc
	000130	script	db200130.scr
	000140	ascii	db200140.asc
	000140	script	db200140.scr
	000150	ascii	db200150.asc
	000150	script	db200150.scr
	000190	ascii	db200190.asc
	000190	script	db200190.scr

Tabla IN_TRAY

Nombre:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
Tipo:	timestamp	char(8)	char(64)	varchar(3000)
Desc:	Date and Time received	User id of person sending note	Brief description	The note

Tabla ORG

Nombre:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
Tipo:	smallint not null	varchar(14)	smallint	varchar(10)	varchar(13)
Desc:	Department number	Department name	Manager number	Division of corporation	City
Valores:	10	Head Office	160	Corporate	New York
	15	New England	50	Eastern	Boston
	20	Mid Atlantic	10	Eastern	Washington
	38	South Atlantic	30	Eastern	Atlanta
	42	Great Lakes	100	Midwest	Chicago
	51	Plains	140	Midwest	Dallas
	66	Pacific	270	Western	San Francisco
	84	Mountain	290	Western	Denver

Tabla PROJECT

Nombre:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
Tipo:	char(6) not null	varchar(24) not null	char(3) not null	char(6) not null	dec(5,2)	date	date	char(6)
Desc:	Project number	Project name	Department responsible	Employee responsible	Estimated mean staffing	Estimated start date	Estimated end date	Major project, for a subproject
Valores:	AD3100	ADMIN SERVICES	D01	000010	6.5	1982-01-01	1983-02-01	-
	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6	1982-01-01	1983-02-01	AD3100
	AD3111	PAYROLL PROGRAMMING	D21	000230	2	1982-01-01	1983-02-01	AD3110
	AD3112	PERSONNEL PROGRAMMING	D21	000250	1	1982-01-01	1983-02-01	AD3110
	AD3113	ACCOUNT PROGRAMMING	D21	000270	2	1982-01-01	1983-02-01	AD3110
	IF1000	QUERY SERVICES	C01	000030	2	1982-01-01	1983-02-01	-
	IF2000	USER EDUCATION	C01	000030	1	1982-01-01	1983-02-01	-
	MA2100	WELD LINE AUTOMATION	D01	000010	12	1982-01-01	1983-02-01	-
	MA2110	W L PROGRAMMING	D11	000060	9	1982-01-01	1983-02-01	MA2100
	MA2111	W L PROGRAM DESIGN	D11	000220	2	1982-01-01	1982-12-01	MA2110
	MA2112	W L ROBOT DESIGN	D11	000150	3	1982-01-01	1982-12-01	MA2110
	MA2113	W L PROD CONT PROGS	D11	000160	3	1982-02-15	1982-12-01	MA2110
	OP1000	OPERATION SUPPORT	E01	000050	6	1982-01-01	1983-02-01	-
	OP1010	OPERATION	E11	000090	5	1982-01-01	1983-02-01	OP1000
	OP2000	GEN SYSTEMS SERVICES	E01	000050	5	1982-01-01	1983-02-01	-
	OP2010	SYSTEMS SUPPORT	E21	000100	4	1982-01-01	1983-02-01	OP2000
	OP2011	SCP SYSTEMS SUPPORT	E21	000320	1	1982-01-01	1983-02-01	OP2010
	OP2012	APPLICATIONS SUPPORT	E21	000330	1	1982-01-01	1983-02-01	OP2010
	OP2013	DB/DC SUPPORT	E21	000340	1	1982-01-01	1983-02-01	OP2010
	PL2100	WELD LINE PLANNING	B01	000020	1	1982-01-01	1982-09-15	MA2100

Tabla SALES

Nombre:	SALES_DATE	SALES_PERSON	REGION	SALES
Tipo:	date	varchar(15)	varchar(15)	int
Desc:	Date of sales	Employee's last name	Region of sales	Number of sales
Valores:	12/31/1995	LUCCHESSI	Ontario-South	1
	12/31/1995	LEE	Ontario-South	3
	12/31/1995	LEE	Quebec	1
	12/31/1995	LEE	Manitoba	2
	12/31/1995	GOUNOT	Quebec	1

Tabla SALES

Nombre:	SALES_DATE	SALES_PERSON	REGION	SALES
	03/29/1996	LUCCHESSI	Ontario-South	3
	03/29/1996	LUCCHESSI	Quebec	1
	03/29/1996	LEE	Ontario-South	2
	03/29/1996	LEE	Ontario-North	2
	03/29/1996	LEE	Quebec	3
	03/29/1996	LEE	Manitoba	5
	03/29/1996	GOUNOT	Ontario-South	3
	03/29/1996	GOUNOT	Quebec	1
	03/29/1996	GOUNOT	Manitoba	7
	03/30/1996	LUCCHESSI	Ontario-South	1
	03/30/1996	LUCCHESSI	Quebec	2
	03/30/1996	LUCCHESSI	Manitoba	1
	03/30/1996	LEE	Ontario-South	7
	03/30/1996	LEE	Ontario-North	3
	03/30/1996	LEE	Quebec	7
	03/30/1996	LEE	Manitoba	4
	03/30/1996	GOUNOT	Ontario-South	2
	03/30/1996	GOUNOT	Quebec	18
	03/30/1996	GOUNOT	Manitoba	1
	03/31/1996	LUCCHESSI	Manitoba	1
	03/31/1996	LEE	Ontario-South	14
	03/31/1996	LEE	Ontario-North	3
	03/31/1996	LEE	Quebec	7
	03/31/1996	LEE	Manitoba	3
	03/31/1996	GOUNOT	Ontario-South	2
	03/31/1996	GOUNOT	Quebec	1
	04/01/1996	LUCCHESSI	Ontario-South	3
	04/01/1996	LUCCHESSI	Manitoba	1
	04/01/1996	LEE	Ontario-South	8
	04/01/1996	LEE	Ontario-North	-
	04/01/1996	LEE	Quebec	8
	04/01/1996	LEE	Manitoba	9
	04/01/1996	GOUNOT	Ontario-South	3
	04/01/1996	GOUNOT	Ontario-North	1
	04/01/1996	GOUNOT	Quebec	3
	04/01/1996	GOUNOT	Manitoba	7

Tabla STAFF

Nombre:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
Tipo:	smallint not null	varchar(9)	smallint	char(5)	smallint	dec(7,2)	dec(7,2)
Desc:	Employee number	Employee name	Department number	Job type	Years of service	Current salary	Commission
Valores:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25

Nombre:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10
	180	Abrahams	38	Clerk	3	12009.75	236.50
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

Tabla STAFFG (sólo para páginas de códigos de doble byte)

Nombre:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
Tipo:	smallint not null	vargraphic(9)	smallint	graphic(5)	smallint	dec(9,0)	dec(9,0)
Desc:	Employee number	Employee name	Department number	Job type	Years of service	Current salary	Commission
Valores:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70

Tabla STAFFG (sólo para páginas de códigos de doble byte)

Nombre:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10
	180	Abrahams	38	Clerk	3	12009.75	236.50
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

Archivos de muestra con tipos de datos BLOB y CLOB

Esta sección muestra los datos encontrados en los archivos EMP_PHOTO (fotos de empleados) y en los archivos EMP_RESUME (resúmenes de empleados).

Foto de Quintana

Figura 14. Dolores M. Quintana

Currículum vitae de Quintana

El texto siguiente se encuentra en los archivos db200130.asc y db200130.scr.

Resume: Dolores M. Quintana

Personal Information

Address:	1150 Eglinton Ave Mellonville, Idaho 83725
Phone:	(208) 555-9933
Birthdate:	September 15, 1925
Sex:	Female
Marital Status:	Married
Height:	5′2″
Weight:	120 lbs.

Department Information

Employee Number:	000130
Dept Number:	C01
Manager:	Sally Kwan
Position:	Analyst
Phone:	(208) 555-4578
Hire Date:	1971-07-28

Education

1965	Math and English, B.A. Adelphi University
1960	Dental Technician Florida Institute of Technology

Work History

10/91 - present	Advisory Systems Analyst Producing documentation tools for engineering department.
12/85 - 9/91	Technical Writer, Writer, text programmer, and planner.
1/79 - 11/85	COBOL Payroll Programmer Writing payroll programs for a diesel fuel company.

Interests

- Cooking
- Reading
- Sewing
- Remodeling

Foto de Nicholls

Figura 15. Heather A. Nicholls

Currículum vitae de Nicholls

El texto siguiente se encuentra en los archivos db200140.asc y db200140.scr.

Resume: Heather A. Nicholls

Personal Information	
Address:	844 Don Mills Ave Mellonville, Idaho 83734
Phone:	(208) 555-2310
Birthdate:	January 19, 1946
Sex:	Female
Marital Status:	Single
Height:	5’8″
Weight:	130 lbs.
Department Information	
Employee Number:	000140
Dept Number:	C01
Manager:	Sally Kwan
Position:	Analyst
Phone:	(208) 555-1793
Hire Date:	1976-12-15
Education	
1972	Computer Engineering, Ph.D. University of Washington
1969	Music and Physics, M.A. Vassar College
Work History	
2/83 - present	Architect, OCR Development Designing the architecture of OCR products.

12/76 - 1/83	Text Programmer Optical character recognition (OCR) programming in PL/I.
9/72 - 11/76	Punch Card Quality Analyst Checking punch cards met quality specifications.

Interests

- Model railroading
- Interior decorating
- Embroidery
- Knitting

Foto de Adamson

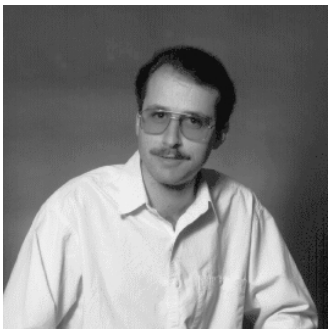


Figura 16. Bruce Adamson

Currículum vitae de Adamson

El texto siguiente se encuentra en los archivos db200150.asc y db200150.scr.

Resume: Bruce Adamson

Personal Information

Address:	3600 Steeles Ave Mellonville, Idaho 83757
Phone:	(208) 555-4489
Birthdate:	May 17, 1947
Sex:	Male
Marital Status:	Married
Height:	6′0″
Weight:	175 lbs.

Department Information

Employee Number:	000150
Dept Number:	D11
Manager:	Irving Stern
Position:	Designer
Phone:	(208) 555-4510
Hire Date:	1972-02-12

Education

1971	Environmental Engineering, M.Sc. Johns Hopkins University
1968	American History, B.A. Northwestern University

Work History

8/79 - present	Neural Network Design Developing neural networks for machine intelligence products.
2/72 - 7/79	Robot Vision Development Developing rule-based systems to emulate sight.
9/71 - 1/72	Numerical Integration Specialist Helping bank systems communicate with each other.

Interests

- Racing motorcycles
- Building loudspeakers
- Assembling personal computers
- Sketching

Foto de Walker

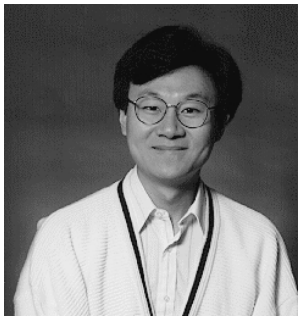


Figura 17. James H. Walker

Currículum vitae de Walker

El texto siguiente se encuentra en los archivos db200190.asc y db200190.scr.

Resume: James H. Walker

Personal Information

Address:	3500 Steeles Ave Mellonville, Idaho 83757
Phone:	(208) 555-7325
Birthdate:	June 25, 1952
Sex:	Male
Marital Status:	Single
Height:	5'11"
Weight:	166 lbs.

Department Information

Employee Number:	000190
Dept Number:	D11
Manager:	Irving Stern
Position:	Designer
Phone:	(208) 555-2986
Hire Date:	1974-07-26

Education

1974	Computer Studies, B.Sc. University of Massachusetts
1972	Linguistic Anthropology, B.A. University of Toronto

Work History

6/87 - present	Microcode Design Optimizing algorithms for mathematical functions.
4/77 - 5/87	Printer Technical Support Installing and supporting laser printers.
9/74 - 3/77	Maintenance Programming Patching assembly language compiler for mainframes.

Interests

- Wine tasting
- Skiing
- Swimming
- Dancing

Currículum vitae de Walker

Apéndice G. Nombres de esquema reservados y palabras reservadas

Existen restricciones sobre el uso de ciertos nombres que el gestor de bases de datos necesita. En algunos casos, los nombres están reservados y los programas de aplicación no pueden utilizarlos. En otros casos, no es aconsejable la utilización de algunos nombres en programas de aplicación, aunque el gestor de bases de datos no impida su utilización.

Los nombres de esquema reservados son los siguientes:

- SYSCAT
- SYSFUN
- SYSIBM
- SYSSTAT
- SYSPROC

Se recomienda encarecidamente que los nombres de esquema no empiecen nunca por el prefijo 'SYS', ya que, por convenio, 'SYS' se utiliza para indicar un área reservada por el sistema. Las funciones no definidas por el usuario, los tipos definidos por el usuario, los activadores o los seudónimos se pueden colocar en un esquema cuyo nombre empiece por 'SYS' (SQLSTATE 42939).

El esquema DB2QP y el esquema SYSTOOLS se ponen a parte para que los utilicen las herramientas de DB2. No es aconsejable que los usuarios definan explícitamente objetos en estos esquemas, aunque el gestor de bases de datos no impida su utilización.

También se recomienda que no se utilice SESSION como nombre de esquema. Como las tablas temporales declaradas deben estar calificadas por SESSION, es posible que una aplicación declare una tabla temporal que tenga un nombre idéntico al de una tabla permanente, lo cual puede producir confusión en la lógica del programa. Para evitar esta situación, no utilice el esquema SESSION excepto cuando utilice tablas temporales declaradas.

No hay palabras específicamente reservadas en DB2 Versión 8. Las palabras clave se pueden utilizar como identificadores normales, excepto en un contexto en que también se interpretarían como palabras clave SQL. En dichos casos, la palabra debe especificarse como un identificador delimitado. Por ejemplo, COUNT no se puede utilizar como un nombre de columna en una sentencia SELECT, a menos que esté delimitado.

ISO/ANSI SQL99 y otros productos DB2 Universal Database incluyen palabras reservadas que, aunque DB2 Universal Database no las imponga, se recomienda que no se utilicen como identificadores normales, ya que esto reduce la portabilidad.

Por motivos de portabilidad entre productos DB2 Universal Database, las siguientes palabras se deben considerar palabras reservadas:

```
ADD              DETERMINISTIC  LEAVE         RESTART
AFTER            DISALLOW       LEFT          RESTRICT
ALIAS            DISCONNECT     LIKE          RESULT
ALL              DISTINCT       LINKTYPE      RESULT_SET_LOCATOR
```

Nombres de esquema reservados y palabras reservadas

ALLOCATE	DO	LOCAL	RETURN
ALLOW	DOUBLE	LOCALE	RETURNS
ALTER	DROP	LOCATOR	REVOKE
AND	DSNHATTR	LOCATORS	RIGHT
ANY	DSSIZE	LOCK	ROLLBACK
APPLICATION	DYNAMIC	LOCKMAX	ROUTINE
AS	EACH	LOCKSIZE	ROW
ASSOCIATE	EDITPROC	LONG	ROWS
ASUTIME	ELSE	LOOP	RRN
AUDIT	ELSEIF	MAXVALUE	RUN
AUTHORIZATION	ENCODING	MICROSECOND	SAVEPOINT
AUX	END	MICROSECONDS	SCHEMA
AUXILIARY	END-EXEC	MINUTE	SCRATCHPAD
BEFORE	END-EXEC1	MINUTES	SECOND
BEGIN	ERASE	MINVALUE	SECONDS
BETWEEN	ESCAPE	MODE	SECQTY
BINARY	EXCEPT	MODIFIES	SECURITY
BUFFERPOOL	EXCEPTION	MONTH	SELECT
BY	EXCLUDING	MONTHS	SENSITIVE
CACHE	EXECUTE	NEW	SET
CALL	EXISTS	NEW_TABLE	SIGNAL
CALLED	EXIT	NO	SIMPLE
CAPTURE	EXTERNAL	NOCACHE	SOME
CARDINALITY	FENCED	NOCYCLE	SOURCE
CASCADED	FETCH	NODENAME	SPECIFIC
CASE	FIELDPROC	NODENUMBER	SQL
CAST	FILE	NOMAXVALUE	SQLID
CCSID	FINAL	NOMINVALUE	STANDARD
CHAR	FOR	NOORDER	START
CHARACTER	FOREIGN	NOT	STATIC
CHECK	FREE	NULL	STAY
CLOSE	FROM	NULLS	STOGROUP
CLUSTER	FULL	NUMPARTS	STORES
COLLECTION	FUNCTION	OBID	STYLE
COLLID	GENERAL	OF	SUBPAGES
COLUMN	GENERATED	OLD	SUBSTRING
COMMENT	GET	OLD_TABLE	SYNONYM
COMMIT	GLOBAL	ON	SYSFUN
CONCAT	GO	OPEN	SYSIBM
CONDITION	GOTO	OPTIMIZATION	SYSPROC
CONNECT	GRANT	OPTIMIZE	SYSTEM
CONNECTION	GRAPHIC	OPTION	TABLE
CONSTRAINT	GROUP	OR	TABLESPACE
CONTAINS	HANDLER	ORDER	THEN
CONTINUE	HAVING	OUT	TO
COUNT	HOLD	OUTER	TRANSACTION
COUNT_BIG	HOUR	OVERRIDING	TRIGGER
CREATE	HOURS	PACKAGE	TRIM
CROSS	IDENTITY	PARAMETER	TYPE
CURRENT	IF	PART	UNDO
CURRENT_DATE	IMMEDIATE	PARTITION	UNION
CURRENT_LC_CTYPE	IN	PATH	UNIQUE
CURRENT_PATH	INCLUDING	PIECESIZE	UNTIL
CURRENT_SERVER	INCREMENT	PLAN	UPDATE
CURRENT_TIME	INDEX	POSITION	USAGE
CURRENT_TIMESTAMP	INDICATOR	PRECISION	USER
CURRENT_TIMEZONE	INHERIT	PREPARE	USING
CURRENT_USER	INNER	PRIMARY	VALIDPROC
CURSOR	INOUT	PRIQTY	VALUES
CYCLE	INSENSITIVE	PRIVILEGES	VARIABLE
DATA	INSERT	PROCEDURE	VARIANT
DATABASE	INTEGRITY	PROGRAM	VCAT
DAY	INTO	PSID	VIEW
DAYS	IS	QUERYNO	VOLUMES
DB2GENERAL	ISOBID	READ	WHEN
DB2GENRL	ISOLATION	READS	WHERE
DB2SQL	ITERATE	RECOVERY	WHILE

DBINFO	JAR	REFERENCES	WITH
DECLARE	JAVA	REFERENCING	WLM
DEFAULT	JOIN	RELEASE	WRITE
DEFAULTS	KEY	RENAME	YEAR
DEFINITION	LABEL	REPEAT	YEARS
DELETE	LANGUAGE	RESET	
DESCRIPTOR	LC_CTYPE	RESIGNAL	

La lista siguiente contiene las palabras reservadas de ISO/ANSI SQL99 que no se encuentran en la lista anterior:

ABSOLUTE	DESCRIBE	MODULE	SESSION
ACTION	DESTROY	NAMES	SESSION_USER
ADMIN	DESTRUCTOR	NATIONAL	SETS
AGGREGATE	DIAGNOSTICS	NATURAL	SIZE
ARE	DICTIONARY	NCHAR	SMALLINT
ARRAY	DOMAIN	NCLOB	SPACE
ASC	EQUALS	NEXT	SPECIFICTYPE
ASSERTION	EVERY	NONE	SQLEXCEPTION
AT	EXEC	NUMERIC	SQLSTATE
BIT	FALSE	OBJECT	SQLWARNING
BLOB	FIRST	OFF	STATE
BOOLEAN	FLOAT	ONLY	STATEMENT
BOTH	FOUND	OPERATION	STRUCTURE
BREADTH	GROUPING	ORDINALITY	SYSTEM_USER
CASCADE	HOST	OUTPUT	TEMPORARY
CATALOG	IGNORE	PAD	TERMINATE
CLASS	INITIALIZE	PARAMETERS	THAN
CLOB	INITIALLY	PARTIAL	TIME
COLLATE	INPUT	POSTFIX	TIMESTAMP
COLLATION	INT	PREFIX	TIMEZONE_HOUR
COMPLETION	INTEGER	PREORDER	TIMEZONE_MINUTE
CONSTRAINTS	INTERSECT	PRESERVE	TRAILING
CONSTRUCTOR	INTERVAL	PRIOR	TRANSLATION
CORRESPONDING	LARGE	PUBLIC	TREAT
CUBE	LAST	REAL	TRUE
CURRENT_ROLE	LATERAL	RECURSIVE	UNDER
DATE	LEADING	REF	UNKNOWN
DEALLOCATE	LESS	RELATIVE	UNNEST
DEC	LEVEL	ROLE	VALUE
DECIMAL	LIMIT	ROLLUP	VARCHAR
DEFERRABLE	LOCALTIME	SCOPE	VARYING
DEFERRED	LOCALTIMESTAMP	SCROLL	WHENEVER
DEPTH	MAP	SEARCH	WITHOUT
DEREF	MATCH	SECTION	WORK
DESC	MODIFY	SEQUENCE	ZONE

Apéndice H. Interacción de los activadores con las restricciones

Este apéndice describe la interacción de los activadores con las restricciones de referencia y las restricciones de comprobación que pueden ser el resultado de una operación de actualización. La Figura 18 y la descripción asociada son representativas del proceso que se lleva a cabo para una sentencia de SQL que actualiza los datos de la base de datos.

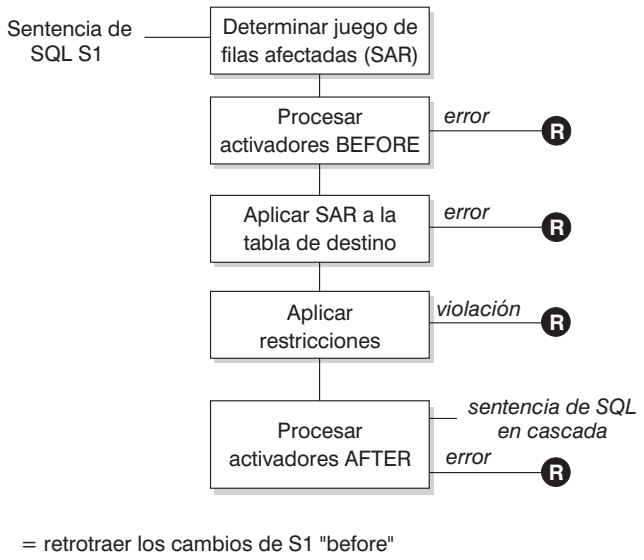


Figura 18. Proceso de una sentencia de SQL con activadores y restricciones asociados

La Figura 18 muestra el orden general de proceso para una sentencia de SQL que actualiza una tabla. Supone una situación donde la tabla incluye activadores anteriores, restricciones de referencia, restricciones de comprobación y activadores posteriores en cascada. A continuación encontrará una descripción de los recuadros y de los demás elementos que se encuentran en la Figura 18.

- Sentencia de SQL S_1

  Es la sentencia DELETE, INSERT o UPDATE que empieza el proceso. La sentencia de SQL S_1 identifica una tabla (o una vista actualizable de alguna tabla) a la que se hace referencia como la *tabla de destino* en toda la descripción.
- Determinación del conjunto de las filas afectadas (*SAR*)

  Este paso es el punto de arranque de un proceso que se repite para las reglas de supresión de restricción de referencia de CASCADE y SET NULL y para las sentencias de SQL en cascada de los activadores posteriores.

  La finalidad de este paso es determinar el *conjunto de filas afectadas* para la sentencia de SQL. El conjunto de filas incluidas en *SAR* se basa en la sentencia:
  - para DELETE, todas las filas que satisfagan la condición de búsqueda de la sentencia (o la fila actual para una DELETE con posición)
  - para INSERT, las filas identificadas por la cláusula VALUES o la selección completa
  - para UPDATE, todas las filas que satisfagan la condición de búsqueda (o la fila actual para una actualización con posición)

Si *SAR* está vacío, no habrá activadores BEFORE, los cambios sólo se aplican a la tabla de destino o a las restricciones para procesar la sentencia de SQL.

- Proceso de activadores BEFORE

  Todos los activadores BEFORE se procesan por orden ascendente de creación. Cada activador BEFORE procesará a acción activada una vez para cada fila de *SAR*.

  Puede producirse un error durante el proceso de una acción activada en cuyo caso se retrotraen todos los cambios realizados como resultado de la sentencia de SQL original S_1 (hasta entonces).

  Si no hay ningún activador BEFORE o si *SAR* está vacío, este paso se salta.

- Aplicación de *SAR* a la tabla de destino

  La supresión, inserción o actualización real se aplica utilizando *SAR* en la tabla de destino de la base de datos.

  Puede producirse un error al aplicar *SAR* (como el intento de insertar una fila con una clave duplicada donde existe un índice de unicidad) en cuyo caso se retrotraen todos los cambios realizados como resultado de la sentencia de SQL original S_1 (hasta entonces).

- Aplicación de restricciones

  Las restricciones asociadas con la tabla de destino se aplican si *SAR* no está vacío. Esto incluye restricciones de unicidad, índices de unicidad, restricciones de referencia, restricciones de comprobación y comprobaciones relacionadas con WITH CHECK OPTION en vistas. Las restricciones de referencia con reglas de supresión en cascada o de establecer nulo pueden provocar que se activen activadores adicionales.

  Una violación de cualquier restricción o WITH CHECK OPTION da como resultado un error y se retrotraen todos los cambios realizados como resultado de S_1 (hasta entonces).

- Proceso de activadores AFTER

  Todos los activadores AFTER activados por S_1 se procesan por orden ascendente de creación.

  Los activadores FOR EACH procesan la acción activada una vez, incluso si *SAR* está vacío. Los activadores FOR EACH ROW procesarán la acción activada una vez para cada fila de *SAR*.

  Puede producirse un error durante el proceso de una acción activada, en cuyo caso se retrotraen todos los cambios realizados como resultado de la S_1 original (hasta entonces).

  La acción activada de un activador puede incluir sentencias de SQL activadas que sean sentencias DELETE, INSERT o UPDATE. Para esta descripción, cada una de dichas sentencias se considera una *sentencia de SQL en cascada*.

  Una sentencia de SQL en cascada es una sentencia DELETE, INSERT o UPDATE que se procesa como parte de la acción activada de un activador AFTER. Esta sentencia empieza un nivel en cascada del proceso de activador. Puede considerarse como la asignación de la sentencia de SQL activada como una S_1 nueva y la realización repetida de todos los pasos descritos aquí.

  Una vez que todas las sentencias de SQL activadas de todos los activadores AFTER activados por cada S_1 hayan terminado su proceso, el proceso de la S_1 original se ha completado.

- **R** = retrotraer cambios antes de S_1

  Cualquier error que se produzca (incluyendo las violaciones de restricciones) durante el proceso da como resultado una retrotracción de todos los cambios realizados directa o indirectamente como resultado de la sentencia de SQL

original S_1. Por lo tanto, la base de datos vuelve a estar en el mismo estado que estaba inmediatamente antes de la ejecución de la sentencia de SQL original S_1

Apéndice I. Tablas de Explain

Tablas de Explain

Las tablas de Explain capturan planes de acceso cuando se activa el recurso Explain. Las tablas de Explain se deben crear antes de invocar Explain. Puede crearlas utilizando las definiciones de tablas documentadas o puede crearlas invocando el script de muestra del procesador de línea de mandatos (CLP), proporcionado en el archivo EXPLAIN.DDL situado en el subdirectorio misc del directorio sqllib. Para invocar el script, conéctese a la base de datos donde se necesitan las tablas de Explain y emita el mandato:

```
db2 -tf EXPLAIN.DDL
```

Cuando el recurso Explain llene las tablas Explain, no activarán activadores ni restricciones de comprobación. Por ejemplo, si se ha definido un activador de inserción en la tabla EXPLAIN_INSTANCE y se ha explicado una sentencia elegible, el activador no se activará.

Para mejorar el rendimiento del recurso Explain en un sistema de bases de datos particionadas, se recomienda crear las tablas de Explain en un solo grupo de particiones de la base de dato particionada, preferiblemente en la misma partición de la base de datos a la que se conectará al compilar la consulta.

Información relacionada:

- "Tabla EXPLAIN_ARGUMENT" en la página 750
- "Tabla EXPLAIN_OBJECT" en la página 756
- "Tabla EXPLAIN_OPERATOR" en la página 759
- "Tabla EXPLAIN_PREDICATE" en la página 761
- "Tabla EXPLAIN_STREAM" en la página 765
- "Tabla ADVISE_INDEX" en la página 767
- "Tabla ADVISE_WORKLOAD" en la página 775
- "Tabla EXPLAIN_INSTANCE" en la página 754
- "Tabla EXPLAIN_STATEMENT" en la página 763
- "Tabla ADVISE_INSTANCE" en la página 770
- "Tabla ADVISE_MQT" en la página 771
- "Tabla ADVISE_PARTITION" en la página 773
- "Tabla ADVISE_TABLE" en la página 774

Tabla EXPLAIN_ARGUMENT

La tabla EXPLAIN_ARGUMENT representa las características exclusivas para cada operador individual, si hay alguno.

Tabla 156. Tabla EXPLAIN_ARGUMENT. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	FK	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	FK	El número de sección en el paquete con el que está relacionada esta información de Explain.
OPERATOR_ID	INTEGER	No	No	ID exclusivo para este operador en esta consulta.
ARGUMENT_TYPE	CHAR(8)	No	No	El tipo de argumento para este operador.
ARGUMENT_VALUE	VARCHAR(1024)	Sí	No	El valor del argumento para este operador. NULL si el valor está en LONG_ARGUMENT_VALUE.
LONG_ARGUMENT_VALUE	CLOB(2M)	Sí	No	El valor del argumento para este operador, cuando el texto no encaja ARGUMENT_VALUE. NULL si el valor está en ARGUMENT_VALUE.

Tabla 157. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	Indicadores de agregación parcial.
BITFLTR	INTEGER FALSE	El tamaño del filtro de bits utilizado por la unión de generación aleatoria.
BLD_LEVEL	Identificador de nivel de creación de DB2	Serie de identificación interna de la versión del código fuente.
BLKLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT SHARE NONE SHARE UPDATE	Intento de bloqueo del nivel de bloqueo.
CSERQY	TRUE FALSE	La consulta remota es una subexpresión común.
CSETEMP	TRUE FALSE	Tabla temporal sobre el Distintivo de subexpresión común.
DIRECT	TRUE	Indicador de lectura directa.
DSTSEVER	Nombre del servidor	Servidor de destino (enviar desde).

Tabla 157. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE (continuación)

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción
DUPLWARN	TRUE FALSE	Duplica el distintivo de Aviso.
EARLYOUT	LEFT RIGHT NONE	Indicador de pronto fuera.
ENVVAR	Cada fila de este tipo contendrá: • Nombre de la variable de entorno • Valor de la variable de entorno	Variable de entorno que afecta al optimizador
FETCHMAX	IGNORE INTEGER	Altera temporalmente el valor del argumento MAXPAGES en el operador FETCH.
GREEDY	TRUE	Indica que el optimizador ha utilizado el algoritmo ávido para planificar el acceso.
GROUPBYC	TRUE FALSE	Indica si se proporcionan columnas Agrupar por.
GROUPBYN	Integer	El número de columnas de comparación.
GROUPBYR	Cada fila de este tipo contendrá: • El valor ordinal de la columna en grupo en clave (seguido por dos puntos y un espacio) • Nombre de columna	Requisito de Agrupar por.
HASHCODE	24 32	Tamaño (en bits) del código de generación aleatoria utilizado para la unión de generación aleatoria.
INNERCOL	Cada fila de este tipo contendrá: • El valor ordinal de la columna en orden (seguido por dos puntos y un espacio) • Nombre de columna • Valor de orden **(A)** Ascendente **(D)** Descendente	Columnas de orden interno.
ISCANMAX	IGNORE INTEGER	Altera temporalmente el valor del argumento MAXPAGES en el operador ISCAN.
JN_INPUT	INNER OUTER	Indica si el operador es el operador que alimenta la parte interna o externa de una unión.
LISTENER	TRUE FALSE	Indicador de Cola de tabla receptora.
MAXPAGES	ALL NONE INTEGER	El número máximo de páginas esperadas para la lectura anticipada.
MAXRIDS	NONE INTEGER	El número máximo de Identificadores de fila que se incluirán en cada petición de lectura anticipada por lista.
NUMROWS	INTEGER	El número de filas que se espera clasificar.
ONEFETCH	TRUE FALSE	Un indicador de lectura.

Tabla 157. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE (continuación)

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción
OUTERCOL	Cada fila de este tipo contendrá: • El valor ordinal de la columna en orden (seguido por dos puntos y un espacio) • Nombre de columna • Valor de orden **(A)** Ascendente **(D)** Descendente	Columnas de orden externo.
OUTERJN	LEFT RIGHT FULL LEFT (ANTI) RIGHT (ANTI)	Indicador de unión externa.
PARTCOLS	Nombre de columna	Columnas de partición para el operador.
PREFETCH	LIST NONE SEQUENTIAL	Tipo de lectura anticipada admisible.
REOPT	ALWAYSONCE	La sentencia se ha optimizado utilizando valores enlazados para los marcadores de parámetros, las variables del lenguaje principal y los registros especiales.
RMTQTEXT	Texto de consulta	Texto de consulta remoto
RNG_PROD	Nombre de función	Función que produce rangos para el acceso ampliado al índice.
ROWLOCK	EXCLUSIVE NONE REUSE SHARE SHORT (INSTANT) SHARE UPDATE	Intento de bloqueo de fila.
ROWWIDTH	INTEGER	Anchura de fila que se ha de clasificar.
RSUFFIX	Texto de consulta	Sufijo de SQL remoto.
SCANDIR	FORWARD REVERSE	Dirección de exploración.
SCANGRAN	INTEGER	Paralelismo intrapartición, granularidad de la exploración paralela de intrapartición, expresada en SCANUNIT.
SCANTYPE	LOCAL PARALLEL	Paralelismo intrapartición, exploración de Índice o Tabla.
SCANUNIT	ROW PAGE	Paralelismo intrapartición, unidad de granularidad de exploración.
SHARED	TRUE	Paralelismo intrapartición, indicador TEMP compartido.
SLOWMAT	TRUE FALSE	Distintivo de materialización lenta.
SNGLPROD	TRUE FALSE	Indicador SORT o TEMP de paralelismo intrapartición generado por un solo agente.

Tabla 157. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE (continuación)

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción
SORTKEY	Cada fila de este tipo contendrá: • El valor ordinal de la columna en clave (seguido por dos puntos y un espacio) • Nombre de columna • Valor de orden **(A)** Ascendente **(D)** Descendente	Columnas de clave de clasificación.
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	Paralelismo intrapartición, tipo SORT.
SRCSEVER	Nombre del servidor	Servidor fuente (enviar a).
SPILLED	INTEGER	El número estimado de páginas del vertido SORT
SQLCA	Información de advertencia	Códigos de advertencia y de razón emitidos durante la operación Explain.
STMTHEAP	INTEGER	Tamaño de la pila de sentencia al iniciar la compilación de una sentencia.
STREAM	TRUE FALSE	La fuente remota es de modalidad continua.
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	Intento de bloqueo de tabla.
TEMPSIZE	INTEGER	Tamaño de página de la tabla temporal.
TQDEGREE	INTEGER	paralelismo intrapartición, número de subagentes que acceden a la Cola de tabla.
TQMERGE	TRUE FALSE	Indicador de Fusión de cola de tabla (clasificada).
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	Propiedad de lectura de Cola de tabla.
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	Propiedad de envío de Cola de tabla.
TQTYPE	LOCAL	Paralelismo intrapartición, Cola de tabla.
TRUNCSRT	TRUE	SORT truncado (limita el número de filas generadas).
UNIQUE	TRUE FALSE	Indicador de exclusividad.
UNIQKEY	Cada fila de este tipo contendrá: • El valor ordinal de la columna en clave (seguido por dos puntos y un espacio) • Nombre de columna	Columnas de claves de unicidad.
VOLATILE	TRUE	Tabla volátil

Tabla EXPLAIN_INSTANCE

La tabla EXPLAIN_INSTANCE es la tabla principal de control para toda la información de Explain. Cada fila de datos de las tablas de Explain se enlaza explícitamente con una fila exclusiva de esta tabla. La tabla EXPLAIN_INSTANCE proporciona la información básica acerca de la fuente de las sentencias de SQL que se están explicando, así como la información acerca del entorno en el que ha tenido lugar la explicación.

Tabla 158. Tabla EXPLAIN_INSTANCE. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	PK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	PK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	PK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	PK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	PK	Versión de la fuente de la petición de Explain.
EXPLAIN_OPTION	CHAR(1)	No	No	Indica que se ha pedido la información de Explain para esta petición. Los posibles valores son: **P** PLAN SELECTION
SNAPSHOT_TAKEN	CHAR(1)	No	No	Indica si se ha tomado una instantánea de Explain para esta petición. Los posibles valores son: **Y** Sí, se ha(n) tomado una(s) instantánea(s) de Explain y se ha(n) almacenado en la tabla EXPLAIN_STATEMENT. También se ha capturado información de Explain normal. **N** No se ha tomado ninguna instantánea de Explain. Se ha capturado información de Explain normal. **O** Sólo se ha tomado una instantánea de Explain. No se ha capturado información de Explain normal.
DB2_VERSION	CHAR(7)	No	No	El número de release del producto para DB2 Universal Database que ha procesado esta petición de Explain. El formato es vv.rr.m, donde: **vv** Número de versión **rr** Número de release **m** Número de release de mantenimiento
SQL_TYPE	CHAR(1)	No	No	Indica si la instancia de Explain era para SQL dinámico o estático. Los posibles valores son: **S** SQL estático **D** SQL dinámico
QUERYOPT	INTEGER	No	No	Indica la clase de optimización de consulta que ha utilizado el Compilador SQL en el momento de la invocación de Explicar. El valor indica qué nivel de optimización de consulta ha efectuado el Compilador SQL para las sentencias de SQL que se explican.

Tabla 158. Tabla EXPLAIN_INSTANCE (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
BLOCK	CHAR(1)	No	No	Indica qué tipo de bloqueo de cursor se ha utilizado al compilar las sentencias de SQL. Para obtener información, consulte la columna BLOCK de SYSCAT.PACKAGES. Los posibles valores son: **N** Sin bloqueo **U** Bloqueo de cursores no ambiguos **B** Bloqueo de todos los cursores
ISOLATION	CHAR(2)	No	No	Indica qué tipo de aislamiento se ha utilizado al compilar las sentencias de SQL. Para obtener más información, consulte la columna ISOLATION en SYSCAT.PACKAGES. Los posibles valores son: **RR** Lectura repetible **RS** Estabilidad de lectura **CS** Estabilidad del cursor **UR** Lectura no confirmada
BUFFPAGE	INTEGER	No	No	Contiene el valor de configuración de base de datos BUFFPAGE en el momento de la invocación de Explicar.
AVG_APPLS	INTEGER	No	No	Contiene el valor del parámetro de configuración AVG_APPLS en el momento de la invocación de Explicar.
SORTHEAP	INTEGER	No	No	Contiene el valor de configuración de la base de datos SORTHEAP en el momento de la invocación de Explicar.
LOCKLIST	INTEGER	No	No	Contiene el valor de configuración de la base de datos LOCKLIST en el momento de la invocación de Explicar.
MAXLOCKS	SMALLINT	No	No	Contiene el valor de configuración de la base de datos MAXLOCKS en el momento de la invocación de Explicar.
LOCKS_AVAIL	INTEGER	No	No	Contiene el número de bloqueos que el optimizador supone que están disponibles para cada usuario. (Derivado de LOCKLIST y MAXLOCKS.)
CPU_SPEED	DOUBLE	No	No	Contiene el valor de configuración de la base de datos CPUSPEED en el momento de la invocación de Explicar.
REMARKS	VARCHAR(254)	Sí	No	Comentario suministrado por el usuario.
DBHEAP	INTEGER	No	No	Contiene el valor de configuración de la base de datos DBHEAP en el momento de la invocación de Explicar.
COMM_SPEED	DOUBLE	No	No	Contiene el valor de configuración de la base de datos COMM_BANDWIDTH en el momento de la invocación de Explicar.
PARALLELISM	CHAR(2)	No	No	Los posibles valores son: • N = Sin paralelismo • P = Paralelismo intrapartición • IP = Paralelismo interparticiones • BP = Paralelismo intrapartición y paralelismo interparticiones
DATAJOINER	CHAR(1)	No	No	Los posibles valores son: • N = Plan de sistemas no federados • Y = Plan de sistemas federados

Tabla EXPLAIN_OBJECT

La tabla EXPLAIN_OBJECT identifica los objetos de datos que necesita el plan de acceso generado para satisfacer la sentencia de SQL.

Tabla 159. Tabla EXPLAIN_OBJECT. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	FK	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	FK	El número de sección en el paquete con el que está relacionada esta información de Explain.
OBJECT_SCHEMA	VARCHAR(128)	No	No	Esquema al que pertenece este objeto.
OBJECT_NAME	VARCHAR(128)	No	No	Nombre del objeto.
OBJECT_TYPE	CHAR(2)	No	No	Etiqueta descriptiva del tipo de objeto.
CREATE_TIME	TIMESTAMP	Sí	No	Hora de creación del objeto; nulo si es una función de tabla.
STATISTICS_TIME	TIMESTAMP	Sí	No	Última vez que se actualizaron las estadísticas para este objeto; nulo si no existen estadísticas para este objeto.
COLUMN_COUNT	SMALLINT	No	No	El número de columnas en este objeto.
ROW_COUNT	INTEGER	No	No	El número estimado de filas en este objeto.
WIDTH	INTEGER	No	No	La anchura media del objeto en bytes. Se establece en -1 para un índice.
PAGES	INTEGER	No	No	El número estimado de páginas que ocupa el objeto en la agrupación de almacenamientos intermedios. Establecido en -1 para una función de tabla.
DISTINCT	CHAR(1)	No	No	Indica si las filas del objeto son diferenciadas (es decir, si hay duplicados) Los valores posibles son: **Y** Sí **N** No
TABLESPACE_NAME	VARCHAR(128)	Sí	No	Nombre del espacio de tablas en el que está almacenado el objeto; se establece en nulo si no está implicado ningún espacio de tablas.
OVERHEAD	DOUBLE	No	No	Actividad general total estimada, en milisegundos, para una sola operación de E/S aleatoria para un espacio de tablas especificado. Incluye la actividad general del controlador, la búsqueda de disco y los tiempos de latencia. Se establece en -1 si no está implicado ningún espacio de tablas.

Tabla 159. Tabla EXPLAIN_OBJECT (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
TRANSFER_RATE	DOUBLE	No	No	Tiempo estimado para leer una página de datos, en milisegundos, del espacio de tablas especificado. Se establece en -1 si no está implicado ningún espacio de tablas.
PREFETCHSIZE	INTEGER	No	No	El número de páginas de datos que se han de leer cuando se efectúa una lectura anticipada. Establecido en -1 para una función de tabla.
EXTENTSIZE	INTEGER	No	No	El tamaño de extensión, en páginas de datos. Este volumen de páginas se escribe en un contenedor individual del espacio de tablas antes de cambiar al contenedor siguiente. Establecido en -1 para una función de tabla.
CLUSTER	DOUBLE	No	No	Nivel de clúster de los datos con el índice. Si >= 1, es el CLUSTERRATIO. Si >= 0 y < 1, es el CLUSTERFACTOR. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
NLEAF	INTEGER	No	No	El número de páginas que ocupan los valores de estos objetos de índice. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
NLEVELS	INTEGER	No	No	El número de niveles de índice del árbol de este objeto de índice. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
FULLKEYCARD	BIGINT	No	No	El número de valores de clave completa diferenciada contenidos en este objeto de índice. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
OVERFLOW	INTEGER	No	No	El número total de registros de desbordamiento en la tabla. Se establece en -1 para un índice, una función de tabla o si no está disponible esta estadística.
FIRSTKEYCARD	BIGINT	No	No	El número de valores de primera clave diferenciada. Se establece en –1 para una tabla, una función de tabla o si no está disponible esta estadística.
FIRST2KEYCARD	BIGINT	No	No	El número de valores de primera clave diferenciada utilizando las primeras columnas {2,3,4} del índice. Se establece en –1 para una tabla, una función de tabla o si no está disponible esta estadística.
FIRST3KEYCARD	BIGINT	No	No	
FIRST4KEYCARD	BIGINT	No	No	
SEQUENTIAL_PAGES	INTEGER	No	No	El número de páginas ubicadas en disco por orden de clave de índice con pocos o ningún vacío entre ellas. Se establece en –1 para una tabla, una función de tabla o si no está disponible esta estadística.
DENSITY	INTEGER	No	No	Proporción de SEQUENTIAL_PAGES en relación al número de páginas del rango de páginas ocupado por el índice, expresada como porcentaje (entero entre 0 y 100). Se establece en –1 para una tabla, una función de tabla o si no está disponible esta estadística.
STATS_SRC	CHAR(1)	No	No	Indica la fuente para las estadísticas. Se establece en 1 si se trata de un solo nodo.
AVERAGE_SEQUENCE_GAP	DOUBLE	No	No	Espacio entre secuencias.
AVERAGE_SEQUENCE_FETCH_GAP	DOUBLE	No	No	Espacio entre las secuencias al captar utilizando el índice.
AVERAGE_SEQUENCE_PAGES	DOUBLE	No	No	Promedio de páginas del índice accesibles en secuencia.
AVERAGE_SEQUENCE_FETCH_PAGES	DOUBLE	No	No	Promedio de páginas de la tabla accesibles en secuencia al captar utilizando el índice.

Tabla 159. Tabla EXPLAIN_OBJECT (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
AVERAGE_RANDOM_PAGES	DOUBLE	No	No	Promedio de páginas del índice aleatorias entre los accesos a páginas secuenciales.
AVERAGE_RANDOM_FETCH_PAGES	DOUBLE	No	No	Promedio de páginas de la tabla aleatorias entre los accesos a páginas secuenciales al captar utilizando el índice.
NUMRIDS	BIGINT	No	No	Número total de identificadores de filas del índice.
NUMRIDS_DELETED	BIGINT	No	No	Número total de identificadores de filas pseudosuprimidos del índice.
NUM_EMPTY_LEAFS	BIGINT	No	No	Número total de páginas hojas vacías del índice.
ACTIVE_BLOCKS	BIGINT	No	No	Número total de bloques de clúster multidimensional (MDC) activos de la tabla.

Tabla 160. Valores de OBJECT_TYPE posibles

Valor	Descripción
IX	Índice
NK	Apodo
RX	Índice RCT
TA	Tabla
TF	Función de tabla
+A	Alias de referencia del compilador
+C	Restricción de referencia del compilador
+F	Función de referencia del compilador
+G	Activador de referencia del compilador
+N	Apodo de referencia del compilador
+T	Tabla de referencia del compilador
+V	Vista de referencia del compilador

Tabla EXPLAIN_OPERATOR

La tabla EXPLAIN_OPERATOR contiene todos los operadores necesarios para que el compilador SQL satisfaga la sentencia de SQL.

Tabla 161. Tabla EXPLAIN_OPERATOR. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	FK	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	FK	El número de sección en el paquete con el que está relacionada esta información de Explain.
OPERATOR_ID	INTEGER	No	No	ID exclusivo para este operador en esta consulta.
OPERATOR_TYPE	CHAR(6)	No	No	Etiqueta descriptiva para el tipo de operador.
TOTAL_COST	DOUBLE	No	No	Coste total acumulado estimado (en timerons) de la ejecución del plan de acceso elegido hasta este operador inclusive.
IO_COST	DOUBLE	No	No	Coste de E/S acumulado estimado (en E/S de páginas de datos) de la ejecución del plan de acceso elegido hasta este operador (inclusive).
CPU_COST	DOUBLE	No	No	Coste de CPU acumulado estimado (en las instrucciones) de la ejecución del plan de acceso elegido hasta este operador (inclusive).
FIRST_ROW_COST	DOUBLE	No	No	Coste acumulado estimado (en timerons) de la lectura de la primera fila para el plan de acceso hasta este operador inclusive. Este valor incluye cualquier actividad general inicial necesaria.
RE_TOTAL_COST	DOUBLE	No	No	Coste acumulado estimado (en timerons) de la lectura de la siguiente fila para el plan de acceso elegido hasta este operador inclusive.
RE_IO_COST	DOUBLE	No	No	Coste de E/S acumulado estimado (en E/S de páginas de datos) de la lectura de la siguiente fila del plan de acceso elegido hasta este operador inclusive.
RE_CPU_COST	DOUBLE	No	No	Coste de CPU acumulado estimado (en instrucciones) de la lectura de la siguiente fila para el plan de acceso elegido hasta este operador inclusive.
COMM_COST	DOUBLE	No	No	Coste de comunicación acumulado estimado (en tramas TCP/IP) de la ejecución del plan de acceso elegido hasta este operador (inclusive).
FIRST_COMM_COST	DOUBLE	No	No	Coste de comunicaciones acumulado estimado (en TCP/IP) de la lectura de la primera fila para el plan de acceso elegido hasta este operador inclusive. Este valor incluye cualquier actividad general inicial necesaria.

Tabla 161. Tabla EXPLAIN_OPERATOR (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
BUFFERS	DOUBLE	No	No	Requisitos estimados de almacenamiento intermedio para este operador y sus entradas.
REMOTE_TOTAL_COST	DOUBLE	No	No	Coste total acumulado estimado (en timerons) de la ejecución de operación(es) en base(s) de datos remota(s).
REMOTE_COMM_COST	DOUBLE	No	No	Coste de comunicación acumulado estimado de la ejecución del plan de acceso remoto elegido hasta este operador (inclusive).

Tabla 162. Valores de OPERATOR_TYPE

Valor	Descripción
DELETE	Suprimir
EISCAN	Exploración de índice ampliada
FETCH	Leer
FILTER	Filtrar filas
GENROW	Generar fila
GRPBY	Agrupar por
HSJOIN	Unión de generación aleatoria
INSERT	Insertar
IXAND	Aplicación de AND de Índice de mapas de bits dinámico
IXSCAN	Exploración de índice
MSJOIN	Fusionar unión de exploración
NLJOIN	Unión de bucle anidado
RETURN	Resultado
RIDSCN	Exploración de identificador de fila (RID)
SHIP	Enviar consulta a sistema remoto
SORT	Clasificar
TBSCAN	Exploración de tabla
TEMP	Construcción de tabla temporal
TQ	Cola de tabla
UNION	Unión
UNIQUE	Eliminación de duplicados
UPDATE	Actualizar

Tabla EXPLAIN_PREDICATE

La tabla EXPLAIN_PREDICATE identifica los predicados que aplica un operador específico.

Tabla 163. Tabla EXPLAIN_PREDICATE. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	FK	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	FK	El número de sección en el paquete con el que está relacionada esta información de Explain.
OPERATOR_ID	INTEGER	No	No	ID exclusivo para este operador en esta consulta.
PREDICATE_ID	INTEGER	No	No	ID exclusivo de este predicado para el operador especificado.
HOW_APPLIED	CHAR(5)	No	No	La forma en que el operador especificado utiliza el predicado.
WHEN_EVALUATED	CHAR(3)	No	No	Indica cuándo se evalúa la subconsulta utilizada en este predicado. Los posibles valores son: **en blanco** Este predicado no contiene ninguna subconsulta. **EAA** La subconsulta utilizada en este predicado se evalúa en la aplicación (EAA). Es decir, se vuelve a evaluar para cada fila procesada por el operador especificado, cuando se aplica el predicado. **EAO** La subconsulta utilizada en este predicado se evalúa en la apertura (EAO). Es decir, se vuelve a evaluar sólo una vez para el operador especificado y sus resultados se vuelven a utilizar en la aplicación del predicado para cada fila. **MUL** Hay más de una subconsulta en este predicado.
RELOP_TYPE	CHAR(2)	No	No	El tipo de operador relacional utilizado en este predicado.

Tabla 163. Tabla EXPLAIN_PREDICATE (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
SUBQUERY	CHAR(1)	No	No	Si es necesaria una corriente de datos de una subconsulta o no para este predicado. Puede ser necesarias múltiples corrientes de subconsultas. Los posibles valores son: **N** No es necesaria ninguna corriente de subconsulta **Y** Son necesarias una o varias corrientes de subconsultas
FILTER_FACTOR	DOUBLE	No	No	La fracción estimada de filas que este predicado calificará.
PREDICATE_TEXT	CLOB(2M)	Sí	No	El texto del predicado tal como se ha vuelto a crear a partir de la representación interna de la sentencia de SQL. Si se utiliza el valor de una variable del lenguaje principal un registro especial o un marcador de parámetros durante la compilación de la sentencia, este valor aparecerá al final del texto del predicado, entre paréntesis. Este valor sólo se almacenará en la tabla EXPLAIN_PREDICATE si la sentencia la ejecuta un usuario que cuente con autorización DBADM o si la variable de registro de DB2 DB2_VIEW_REOPT_VALUES está definida en YES; en caso contrario, aparecerán unos paréntesis vacíos al final del texto del predicado. Nulo si no está disponible.

Tabla 164. Valores de HOW_APPLIED posibles

Valor	Descripción
BSARG	Evaluado como un predicado comparable una vez para cada bloque
JOIN	Utilizado para unir tablas
RESID	Evaluado como un predicado residual
SARG	Evaluado como un predicado comparable para un índice o página de datos
START	Utilizado como una condición de inicio
STOP	Utilizado como una condición de detención

Tabla 165. Valores de RELOP_TYPE posibles

Valor	Descripción
blancos	No aplicable
EQ	Igual
GE	Mayor o igual que
GT	Mayor que
IN	En lista
LE	Menor o igual que
LK	Igual
LT	Menor que
NE	Diferente a
NL	Es nulo
NN	No es nulo

Tabla EXPLAIN_STATEMENT

La tabla EXPLAIN_STATEMENT contiene el texto de la sentencia de SQL tal como existe para los diferentes niveles de información de Explain. La sentencia de SQL original se almacena tal como la entra el usuario, en esta tabla junto con la versión utilizada (por el optimizador) para elegir el plan de acceso para satisfacer la sentencia de SQL. La última versión puede parecerse poco a la original ya que puede haberse vuelto a escribir y/o mejorar con predicados adicionales tal como lo determina el Compilador SQL.

Tabla 166. Tabla EXPLAIN_STATEMENT. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	PK, FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	PK, FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	PK, FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	PK, FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	PK	Nivel de información de Explain para el que esta fila es aplicable. Los valores válidos son: **O** Texto original (tal como lo ha entrado el usuario) **P** PLAN SELECTION
STMTNO	INTEGER	No	PK	El número de sentencia en el paquete con el que está relacionado esta información de Explain. Establecido en 1 para las sentencias de SQL dinámico de Explain. Para las sentencias de SQL estático, este valor es igual al valor utilizado para la vista de catálogo SYSCAT.STATEMENTS.
SECTNO	INTEGER	No	PK	El número de sección en el paquete que contiene esta sentencia de SQL. En las sentencias de SQL dinámico de Explain, es el número de sección utilizada para contener la sección para esta sentencia en tiempo de ejecución. Para las sentencias de SQL estático, este valor es igual al valor utilizado para la vista de catálogo SYSCAT.STATEMENTS.
QUERYNO	INTEGER	No	No	Identificador numérico para la sentencia de SQL explicada. Para sentencias de SQL dinámico (excluyendo la sentencia EXPLAIN SQL) emitidas a través de CLP o CLI, el valor por omisión es un valor incrementado secuencialmente. De lo contrario, el valor por omisión es el valor de STMTNO para sentencias de SQL estático y 1 para sentencias de SQL dinámico.
QUERYTAG	CHAR(20)	No	No	Distintivo identificador para cada sentencia de SQL explicada. Para sentencias de SQL dinámico emitidas a través de CLP (excluyendo la sentencia EXPLAIN SQL), el valor por omisión es 'CLP'. Para sentencias de SQL dinámico emitidas a través de CLI (excluyendo la sentencia EXPLAIN SQL), el valor por omisión es 'CLI'. De lo contrario, el valor por omisión utilizado es blancos.

Tabla 166. Tabla EXPLAIN_STATEMENT (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
STATEMENT_TYPE	CHAR(2)	No	No	Etiqueta descriptiva para el tipo de consulta que se está explicando. Los posibles valores son: **S** Selección **D** Suprimir **DC** Supresión en la ubicación actual del cursor **I** Insertar **U** Actualizar **UC** Actualización en la ubicación actual del cursor
UPDATABLE	CHAR(1)	No	No	Indica si esta sentencia se considera actualizable. Es relevante en particular para las sentencias SELECT que pueden determinarse como actualizables en potencia. Los posibles valores son: **' '** No aplicable (blanco) **N** No **Y** Sí
DELETABLE	CHAR(1)	No	No	Indica si esta sentencia se considera suprimible. Es relevante en particular para las sentencias SELECT que pueden determinarse como suprimibles en potencia. Los posibles valores son: **' '** No aplicable (blanco) **N** No **Y** Sí
TOTAL_COST	DOUBLE	No	No	Coste total estimado (en timerons) de la ejecución del plan de acceso elegido para esta sentencia; se establece en 0 (cero) si EXPLAIN_LEVEL es *O* (texto original) ya que no se ha elegido ningún plan de acceso en este momento.
STATEMENT_TEXT	CLOB(2M)	No	No	Texto o fragmento del texto de la sentencia de SQL que se está explicando. El texto que se muestra para el nivel de Explain de Selección del plan se ha reconstruido a partir de la representación interna y es parecido a SQL en su naturaleza; es decir, no se garantiza que la sentencia reconstruida siga la sintaxis SQL correcta.
SNAPSHOT	BLOB(10M)	Sí	No	Instantánea de la representación interna de esta sentencia de SQL en el Nivel_Explain mostrado. Esta columna está pensada para utilizarla con DB2 Visual Explain. La columna se establece en nulo si EXPLAIN_LEVEL es *0* (sentencia original), pues no se había elegido ningún plan de acceso todavía en el momento en que se capturó esta versión específica de la sentencia.
QUERY_DEGREE	INTEGER	No	No	Indica el grado de paralelismo intrapartición en el momento de la invocación de Explicar. Para la sentencia original, contiene el grado dirigido de paralelismo intrapartición. Para PLAN SELECTION, contiene el grado de paralelismo intrapartición generado para que lo utilice el plan.

Tabla EXPLAIN_STREAM

La tabla EXPLAIN_STREAM representa las corrientes de datos de entrada y de salida entre operadores individuales y objetos de datos. Los objetos de datos en sí se representan en la tabla EXPLAIN_OBJECT. Los operadores implicados en una corriente de datos se han de encontrar en una tabla EXPLAIN_OPERATOR.

Tabla 167. Tabla EXPLAIN_STREAM. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	FK	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	FK	El número de sección en el paquete con el que está relacionada esta información de Explain.
STREAM_ID	INTEGER	No	No	ID exclusivo para esta corriente de datos en el operador especificado.
SOURCE_TYPE	CHAR(1)	No	No	Indica la fuente de la corriente de datos: **O** Operador **D** Objeto de datos
SOURCE_ID	SMALLINT	No	No	ID exclusivo para el operador dentro de esta consulta que es la fuente de esta corriente de datos. Se establece en -1 si SOURCE_TYPE es 'D'.
TARGET_TYPE	CHAR(1)	No	No	Indica el destino de la corriente de datos: **O** Operador **D** Objeto de datos
TARGET_ID	SMALLINT	No	No	ID exclusivo para el operador dentro de esta consulta que es el destino de esta corriente de datos. Se establece en -1 si TARGET_TYPE es 'D'.
OBJECT_SCHEMA	VARCHAR(128)	Sí	No	El esquema al que pertenece el objeto de datos afectado. Se establece en nulo si SOURCE_TYPE y TARGET_TYPE son 'O'.
OBJECT_NAME	VARCHAR(128)	Sí	No	Nombre del objeto que es el sujeto de la corriente de datos. Se establece en nulo si SOURCE_TYPE y TARGET_TYPE son 'O'.
STREAM_COUNT	DOUBLE	No	No	Cardinalidad estimada de la corriente de datos.
COLUMN_COUNT	SMALLINT	No	No	El número de columnas en la corriente de datos.
PREDICATE_ID	INTEGER	No	No	Si la corriente forma parte de una subconsulta para un predicado, el ID del predicado se reflejará aquí, de lo contrario la columna se establece en -1.

Tabla 167. Tabla EXPLAIN_STREAM (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
COLUMN_NAMES	CLOB(2M)	Sí	No	Esta columna contiene los nombres y la información de ordenación de las columnas implicadas en esta corriente. Estos nombres estarán en el formato de: `NOMBRE1(A)+NOMBRE2(D)+NOMBRE3+NOMBRE4` Donde *(A)* indica una columna por orden ascendente, *(D)* indica una columna por orden descendente y ninguna información de ordenación indica que la columna no está ordenada o que el orden no es relevante.
PMID	SMALLINT	No	No	ID del mapa de particionamiento.
SINGLE_NODE	CHAR(5)	Sí	No	Indica si este flujo de datos está en una partición única o en varias particiones: **MULT** En varias particiones **COOR** En nodo del coordinador **HASH** Dirigido utilizando generación aleatoria **RID** Dirigido utilizando el ID de fila **FUNC** Dirigido utilizando una función (HASHEDVALUE() o DBPARTITIONNUM()) **CORR** Dirigido utilizando un valor de correlación **Numberic** Dirigido hacia un nodo individual predeterminado
PARTITION_COLUMNS	CLOB(2M)	Sí	No	Lista de columnas en las que este flujo de datos está particionado.

Tabla ADVISE_INDEX

La tabla ADVISE_INDEX representa los índices recomendados.

Tabla 168. Tabla ADVISE_INDEX. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	No	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	No	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	No	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	No	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	No	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	No	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	No	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	No	El número de sección en el paquete con el que está relacionada esta información de Explain.
QUERYNO	INTEGER	No	No	Identificador numérico para la sentencia de SQL explicada. Para sentencias de SQL dinámico (excluyendo la sentencia EXPLAIN SQL) emitidas a través de CLP o CLI, el valor por omisión es un valor incrementado secuencialmente. De lo contrario, el valor por omisión es el valor de STMTNO para sentencias de SQL estático y 1 para sentencias de SQL dinámico.
QUERYTAG	CHAR(20)	No	No	Distintivo identificador para cada sentencia de SQL explicada. Para sentencias de SQL dinámico emitidas a través de CLP (excluyendo la sentencia EXPLAIN SQL), el valor por omisión es 'CLP'. Para sentencias de SQL dinámico emitidas a través de CLI (excluyendo la sentencia EXPLAIN SQL), el valor por omisión es 'CLI'. De lo contrario, el valor por omisión utilizado es blancos.
NAME	VARCHAR(128)	No	No	Nombre del índice.
CREATOR	VARCHAR(128)	No	No	Calificador del nombre de índice.
TBNAME	VARCHAR(128)	No	No	Nombre de la tabla o apodo en el que se define el índice.
TBCREATOR	VARCHAR(128)	No	No	Calificador del nombre de tabla.
COLNAMES	CLOB(2M)	No	No	Lista de nombres de columnas.
UNIQUERULE	CHAR(1)	No	No	Regla de unicidad: D = Los duplicados están permitidos P = Índice primario U = Sólo se permiten entradas exclusivas
COLCOUNT	SMALLINT	No	No	El número de columnas de la clave más el número de columnas INCLUDE si hay alguna.
IID	SMALLINT	No	No	ID interno de índice.
NLEAF	INTEGER	No	No	El número de páginas; –1 si no se reúnen estadísticas.
NLEVELS	SMALLINT	No	No	Número de niveles de índices; –1 si no se reúnen estadísticas.

Tabla 168. Tabla ADVISE_INDEX (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
FIRSTKEYCARD	BIGINT	No	No	El número de valores de primera clave diferenciada; –1 si no se reúnen estadísticas.
FULLKEYCARD	BIGINT	No	No	El número de valores de clave completa diferenciada; –1 si no se reúnen estadísticas.
CLUSTERRATIO	SMALLINT	No	No	Nivel de clúster de los datos con el índice; -1 si no se reúnen estadísticas o si se reúnen estadísticas de índice detalladas (en este caso, se utilizará CLUSTERFACTOR en su lugar).
CLUSTERFACTOR	DOUBLE	No	No	Mejor medición del nivel de clúster o –1 si no se han reunido estadísticas de índice detalladas o si el índice está definido en un apodo.
USERDEFINED	SMALLINT	No	No	Definido por el usuario.
SYSTEM_REQUIRED	SMALLINT	No	No	1 si se cumple una de las condiciones siguientes: – Este índice es necesario para una restricción de clave primaria o exclusiva o bien este índice es un índice de bloques de dimensión o un índice de bloques compuestos para una tabla con clústeres de varias dimensiones (MDC). – Se trata de un índice en la columna OID de una tabla con tipo. 2 si se cumplen estas dos condiciones: – Este índice es necesario para una restricción de clave primaria o exclusiva o bien este índice es un índice de bloques de dimensión o un índice de bloques compuestos para una tabla MDC. – Se trata de un índice en la columna OID de una tabla con tipo. De lo contrario, 0.
CREATE_TIME	TIMESTAMP	No	No	Hora en que se ha creado el índice.
STATS_TIME	TIMESTAMP	Sí	No	Última vez que se ha realizado un cambio en las estadísticas registradas para este índice. Nulo si no hay estadísticas disponibles.
PAGE_FETCH_PAIRS	VARCHAR(254)	No	No	Una lista de pares de enteros, representada en la forma de caracteres. Cada par representa el número de páginas de un almacenamiento intermedio hipotético y el número de lecturas de páginas necesario para explorar la tabla con este índice utilizando dicho almacenamiento intermedio hipotético. (Serie de longitud cero si no hay datos disponibles.)
REMARKS	VARCHAR(254)	Sí	No	Comentario suministrado por el usuario o nulo.
DEFINER	VARCHAR(128)	No	No	Usuario que ha creado el índice.
CONVERTED	CHAR(1)	No	No	Reservado para su utilización en el futuro.
SEQUENTIAL_PAGES	INTEGER	No	No	El número de páginas ubicadas en disco por orden de clave de índice con pocos o ningún vacío entre ellas. (–1 si no hay estadísticas disponibles.)
DENSITY	INTEGER	No	No	Proporción de SEQUENTIAL_PAGES para numerar las páginas del rango de páginas ocupadas por el índice, expresada como un porcentaje (entero entre 0 y 100, –1 si no hay estadísticas disponibles.)
FIRST2KEYCARD	BIGINT	No	No	El número de claves diferenciadas que utilizan las dos primeras columnas del índice (–1 si no hay estadísticas o no son aplicables)

Tabla 168. Tabla ADVISE_INDEX (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
FIRST3KEYCARD	BIGINT	No	No	El número de claves diferenciadas que utilizan las tres primeras columnas del índice (–1 si no hay estadísticas o si no son aplicables)
FIRST4KEYCARD	BIGINT	No	No	El número de claves diferenciadas que utilizan las cuatro primeras columnas del índice (–1 si no hay estadísticas o no son aplicables)
PCTFREE	SMALLINT	No	No	Porcentaje de cada página de hoja de índice que se va a reservar durante la creación inicial del índice. Este espacio está disponible para inserciones futuras después de la creación del índice.
UNIQUE_COLCOUNT	SMALLINT	No	No	El número de columnas necesarias para una clave de unicidad. Siempre <=COLCOUNT. < COLCOUNT solamente si hay columnas INCLUDE. –1 si el índice no tiene clave de unicidad (permite duplicados)
MINPCTUSED	SMALLINT	No	No	Si no es cero, se habilita la desfragmentación del índice en línea y el valor es el umbral del espacio mínimo utilizado antes de fusionar las páginas.
REVERSE_SCANS	CHAR(1)	No	No	Y = El índice soporta exploraciones invertidas N = El índice no soporta exploraciones invertidas
USE_INDEX	CHAR(1)	Sí	No	Y = índice recomendado o evaluado N = índice no recomendado R = se ha recomendado la separación del clúster de un índice de clúster RID (por parte del Asesor de diseño); esto ocurre cuando se recomienda un índice de clúster RID nuevo para la tabla.
CREATION_TEXT	CLOB(2M)	No	No	La sentencia de SQL se utiliza para crear el índice.
PACKED_DESC	BLOB(1M)	Sí	No	Descripción interna de la tabla.
RUN_ID	TIMESTAMP	Sí	FK	Valor correspondiente a START_TIME de una fila de la tabla ADVISE_INSTANCE que la enlaza con la misma ejecución del Asesor de diseño.
INDEXTYPE	VARCHAR(4)	No	No	Tipo de índice. CLUS = Clústeres REG = Normal DIM = Índice de bloques de dimensión BLOK = Índice de bloques
EXISTS	CHAR(1)	No	No	Debe definirse en 'Y' si el índice ya existe en el catálogo de la base de datos.
RIDTOBLOCK	CHAR(1)	No	No	Debe definirse en 'Y' si se ha utilizado el índice RID para crear un índice de bloques en el Asesor de diseño.

Tabla ADVISE_INSTANCE

La tabla ADVISE_INSTANCE contiene información acerca de la ejecución de **db2advis**, incluida la hora de inicio. Contiene una fila para cada ejecución de **db2advis**. Otras tablas ADVISE presentan una clave foránea (RUN_ID) que enlaza las filas creadas durante la misma ejecución del Asesor de diseño con la columna START_TIME de la tabla ADVISE_INSTANCE.

Tabla 169. Tabla ADVISE_INSTANCE. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
START_TIME	TIMESTAMP	No	PK	Hora en la que empieza la ejecución de **db2advis**.
END_TIME	TIMESTAMP	No	No	Hora en la que finaliza la ejecución de **db2advis**.
MODE	VARCHAR(4)	No	No	El valor que se ha especificado con la opción -m en el Asesor de diseño; por ejemplo, 'MC' para especificar MQT y MDC.
WKLD_COMPRESSION	CHAR(4)	No	No	La compresión de carga de trabajo bajo la que se ha ejecutado el Asesor de diseño.
STATUS	CHAR(9)	No	No	Estado de una ejecución del Asesor de diseño. El estado puede ser 'STARTED', 'COMPLETED' (si es satisfactorio) o un número de error que lleve el prefijo 'EI' para los errores internos o 'EX' para los errores externos, en cuyo caso el número de error representa el SQLCODE.

Tabla ADVISE_MQT

La tabla ADVISE_MQT contiene información acerca de las tablas de consulta materializadas (MQT) recomendadas por el Asesor de diseño.

Tabla 170. Tabla ADVISE_MQT. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	No	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	No	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	No	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	No	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	No	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	No	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	No	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	No	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
NAME	VARCHAR(128)	No	No	Nombre de la MQT.
CREATOR	VARCHAR(128)	No	No	Nombre del creador de la MQT.
IID	SMALLINT	No	No	Identificador interno.
CREATE_TIME	TIMESTAMP	No	No	Hora en que se ha creado la MQT.
STATS_TIME	TIMESTAMP	Sí	No	Hora en que se han recogido las estadísticas.
NUMROWS	DOUBLE	No	No	Número estimado de filas de la MQT.
NUMCOLS	SMALLINT	No	No	Número de columnas definido en la MQT.
ROWSIZE	DOUBLE	No	No	Longitud media (en bytes) de una fila de la MQT.
BENEFIT	FLOAT	No	No	Reservado para su utilización en el futuro.
USE_MQT	CHAR(1)	Sí	No	Debe definirse en 'Y' si se recomienda la MQT.
MQT_SOURCE	CHAR(1)	Sí	No	Indica si se ha generado el candidato de la MQT. Debe definirse en 'I' si el candidato de la MQT es una MQT de renovación inmediata o en 'D' si sólo puede crearse como una MQT de renovación diferida completa.
QUERY_TEXT	CLOB(2M)	No	No	Contiene la consulta que define la MQT.
CREATION_TEXT	CLOB(2M)	No	No	Contiene el DDL de CREATE TABLE para la MQT.
SAMPLE_TEXT	CLOB(2M)	No	No	Contiene la consulta de muestreo que se utiliza para obtener estadísticas detalladas para la MQT. Sólo se utiliza cuando se necesitan estadísticas detalladas para el Asesor de diseño. Las estadísticas de muestreo resultantes se mostrarán en esta tabla. Si es nulo, no se había creado ninguna consulta de muestreo para esta MQT.
COLSTATS	CLOB(2M)	No	No	Contiene las estadísticas de columna para la MQT (si no es nulo). Estas estadísticas están en formato XML e incluyen el nombre de la columna, la cardinalidad de la columna y, opcionalmente, los valores HIGH2KEY y LOW2KEY.
EXTRA_INFO	BLOB(2M)	No	No	Reservado para salidas varias.
TBSPACE	VARCHAR(128)	No	No	El espacio de tablas que se recomienda para la MQT.

Tabla ADVISE_MQT

Tabla 170. Tabla ADVISE_MQT (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
RUN_ID	TIMESTAMP	Sí	FK	Valor correspondiente a START_TIME de una fila de la tabla ADVISE_INSTANCE que la enlaza con la misma ejecución del Asesor de diseño.
REFRESH_TYPE	CHAR(1)	No	No	Debe definirse en 'I' para inmediato o en 'D' para diferido.
EXISTS	CHAR(1)	No	No	Debe definirse en 'Y' si la MQT ya existe en el catálogo de la base de datos.

Tabla ADVISE_PARTITION

La tabla ADVISE_PARTITION contiene información acerca de las particiones de la base de datos recomendadas por el Asesor de diseño y sólo puede rellenarse en un entorno de bases de datos particionadas.

Tabla 171. Tabla ADVISE_PARTITION. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	No	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	No	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	No	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	No	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	No	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	No	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	No	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	No	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
QUERYNO	INTEGER	No	No	Identificador numérico para la sentencia de SQL explicada. Para sentencias de SQL dinámico (excluyendo la sentencia EXPLAIN SQL) emitidas a través de CLP o CLI, el valor por omisión es un valor incrementado secuencialmente. De lo contrario, el valor por omisión es el valor de STMTNO para sentencias de SQL estático y 1 para sentencias de SQL dinámico.
QUERYTAG	CHAR(20)	No	No	Distintivo identificador para cada sentencia de SQL explicada. Para sentencias de SQL dinámico emitidas a través de CLP (excluida la sentencia EXPLAIN SQL), el valor por omisión es 'CLP'. Para sentencias de SQL dinámico emitidas a través de CLI (excluida la sentencia EXPLAIN SQL), el valor por omisión es 'CLI'. De lo contrario, el valor por omisión utilizado es blancos.
TBNAME	VARCHAR(128)	Sí	No	Especifica el nombre de la tabla.
TBCREATOR	VARCHAR(128)	Sí	No	Especifica el nombre del creador de la tabla.
PMID	SMALLINT	Sí	No	Especifica el ID de correlación de la partición.
TBSPACE	VARCHAR(128)	Sí	No	Especifica el espacio de tablas en el que reside la tabla.
COLNAMES	CLOB(2M)	Sí	No	Especifica los nombres de las columnas de las particiones, separados por comas.
COLCOUNT	SMALLINT	Sí	No	Especifica el número de columnas de particionamiento.
REPLICATE	CHAR(1)	Sí	No	Especifica si la partición se duplica o no se duplica.
COST	DOUBLE	Sí	No	Especifica el coste del uso de la partición.
USEIT	CHAR(1)	Sí	No	Especifica si la partición se utiliza o no en modalidad EVALUATE PARTITION. La partición se utiliza si USEIT está definido en 'Y' o 'y'.
RUN_ID	TIMESTAMP	Sí	FK	Valor correspondiente a START_TIME de una fila de la tabla ADVISE_INSTANCE que la enlaza con la misma ejecución del Asesor de diseño.

Tabla ADVISE_TABLE

La tabla ADVISE_TABLE almacena el lenguaje de definición de datos (DDL) para la creación de tablas utilizando las recomendaciones finales del Asesor de diseño para las tablas de consulta materializadas (MQT), las tablas de clústeres de varias dimensiones (MDC) y el particionamiento.

Tabla 172. Tabla ADVISE_TABLE. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
RUN_ID	TIMESTAMP	Sí	FK	Valor correspondiente a START_TIME de una fila de la tabla ADVISE_INSTANCE que la enlaza con la misma ejecución del Asesor de diseño.
TABLE_NAME	VARCHAR(128)	No	No	Nombre de la tabla.
TABLE_SCHEMA	VARCHAR(128)	No	No	Nombre del creador de la tabla.
TABLESPACE	VARCHAR(128)	No	No	El espacio de tablas en el que va a crearse la tabla.
SELECTION_FLAG	VARCHAR(4)	No	No	Indica el tipo de recomendación. Los valores válidos son 'M' para MQT, 'P' para particionamiento y 'C' para MDC. Este campo puede incluir cualquier subconjunto de estos valores. Por ejemplo, 'MC' indica que la tabla se recomienda como una tabla MQT y MDC.
TABLE_EXISTS	CHAR(1)	No	No	Debe definirse en 'Y' si la tabla ya existe en el catálogo de la base de datos.
USE_TABLE	CHAR(1)	No	No	Debe definirse en 'Y' si la tabla tiene recomendaciones del Asesor de diseño.
GEN_COLUMNS	CLOB(2M)	No	No	Contiene una serie de columnas generadas si esta fila incluye una recomendación de MDC que necesita columnas generadas en el DDL de creación de tablas.
ORGANIZE_BY	CLOB(2M)	No	No	Para las recomendaciones de MDC, contiene la cláusula ORGANIZE BY del DDL de creación de tablas.
CREATION_TEXT	CLOB(2M)	No	No	Contiene el DDL de creación de tablas.
ALTER_COMMAND	CLOB(2M)	No	No	Contiene una sentencia ALTER TABLE para la tabla.

Tabla ADVISE_WORKLOAD

La tabla ADVISE_WORKLOAD representa la sentencia que forma la carga de trabajo.

Tabla 173. Tabla ADVISE_WORKLOAD. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibilidad de nulos?	¿Clave?	Descripción
WORKLOAD_NAME	CHAR(128)	No	No	Nombre del conjunto de sentencias de SQL (carga de trabajo) a la que esta sentencia pertenece.
STATEMENT_NO	INTEGER	No	No	El número de sentencias de la carga de trabajo con el que está relacionada esta información de explicación.
STATEMENT_TEXT	CLOB(1M)	No	No	Contenido de la sentencia de SQL.
STATEMENT_TAG	VARCHAR(256)	No	No	Distintivo identificador para cada sentencia de SQL explicada.
FREQUENCY	INTEGER	No	No	Las veces que esta sentencia aparece en la carga de trabajo.
IMPORTANCE	DOUBLE	No	No	Importancia de la sentencia.
WEIGHT	DOUBLE	No	No	Prioridad de la sentencia.
COST_BEFORE	DOUBLE	Sí	No	El coste (en timerons) de la consulta si no se crean los índices recomendados.
COST_AFTER	DOUBLE	Sí	No	El coste (en timerons) de la consulta si se crean los índices recomendados.
COMPILABLE	CHAR(17)	Sí	No	Indica los errores de compilación de la consulta que se han producido al intentar preparar la sentencia. Si esta columna es NULL o no empieza por SQLCA, la consulta de SQL podría compilarla db2advis. Si db2advis o el Asesor de diseño detectan un error de compilación, el valor de columna COMPILABLE constará de un campo SQLCA.sqlcaid de 8 caracteres seguido de dos puntos (:) y un campo SQLCA.sqlstate de 8 caracteres, que es el código de retorno de la sentencia SQL.

Apéndice J. Valores de los registros de EXPLAIN

A continuación se proporciona una descripción de la interacción de los valores de los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT, entre sí y con los mandatos PREP y BIND.

Con SQL dinámico, los valores de los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT interactúan de la siguiente manera:

Tabla 174. Interacción de los valores de los registros especiales EXPLAIN (SQL dinámico)

Valores de EXPLAIN SNAPSHOT	Valores de EXPLAIN MODE					
	NO	**YES**	**EXPLAIN**	**REOPT**	**RECOMMEND INDEXES**	**EVALUATE INDEXES**
NO	• Se devuelven los resultados de la consulta.	• Se rellenan las tablas de explicación. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de explicación se rellenan cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Se rellenan las tablas de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.
YES	• Se toma una Instantánea de explicación. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de explicación. • Se toma una Instantánea de explicación. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de explicación. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de explicación se rellenan cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de explicación. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Se rellenan las tablas de explicación. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.

Tabla 174. Interacción de los valores de los registros especiales EXPLAIN (SQL dinámico) (continuación)

Valores de EXPLAIN SNAPSHOT	Valores de EXPLAIN MODE					
	NO	**YES**	**EXPLAIN**	**REOPT**	**RECOMMEND INDEXES**	**EVALUATE INDEXES**
EXPLAIN	• Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Se rellenan las tablas de explicación. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Se rellenan las tablas de explicación. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de explicación se rellenan cuando una sentencia está calificada para su reoptimización durante la ejecución. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas o de vinculación incremental no se ejecutan).	• Se rellenan las tablas de explicación. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Se rellenan las tablas de explicación. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.

Tabla 174. Interacción de los valores de los registros especiales EXPLAIN (SQL dinámico) (continuación)

Valores de EXPLAIN SNAPSHOT	Valores de EXPLAIN MODE					
	NO	**YES**	**EXPLAIN**	**REOPT**	**RECOMMEND INDEXES**	**EVALUATE INDEXES**
REOPT	• Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de explicación. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de explicación. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas o de vinculación incremental no se ejecutan).	• Las tablas de explicación se rellenan cuando una sentencia está calificada para su reoptimización durante la ejecución. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de explicación. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas o de vinculación incremental no se ejecutan). • Se recomiendan los índices.	• Se rellenan las tablas de explicación. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas o de vinculación incremental no se ejecutan). • Se evalúan los índices.

El registro especial CURRENT EXPLAIN MODE interactúa con la opción de vinculación EXPLAIN de la siguiente manera en SQL dinámico.

Tabla 175. Interacción de la opción de vinculación EXPLAIN y CURRENT EXPLAIN MODE

Valores de EXPLAIN MODE	Valores de la opción de vinculación EXPLAIN			
	NO	**YES**	**REOPT**	**ALL**
NO	• Se devuelven los resultados de la consulta.	• Las tablas de explicación se rellenan para SQL estático. • Se devuelven los resultados de la consulta.	• Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Las tablas de explicación se rellenan para SQL estático. • Las tablas de explicación se llenan para SQL dinámico. • Se devuelven los resultados de la consulta.
YES	• Las tablas de explicación se rellenan para SQL dinámico. • Se devuelven los resultados de la consulta.	• Las tablas de explicación se rellenan para SQL estático. • Las tablas de explicación se rellenan para SQL dinámico. • Se devuelven los resultados de la consulta.	• Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Las tablas de explicación se rellenan para SQL estático. • Las tablas de explicación se rellenan para SQL dinámico. • Se devuelven los resultados de la consulta.

Tabla 175. Interacción de la opción de vinculación EXPLAIN y CURRENT EXPLAIN MODE (continuación)

Valores de EXPLAIN MODE	Valores de la opción de vinculación EXPLAIN			
	NO	YES	REOPT	ALL
EXPLAIN	• Las tablas de explicación se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de explicación se rellenan para SQL estático. • Las tablas de explicación se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de explicación se rellenan para SQL estático. • Las tablas de explicación se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).
REOPT	• Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.

Tabla 175. Interacción de la opción de vinculación EXPLAIN y CURRENT EXPLAIN MODE (continuación)

Valores de EXPLAIN MODE	Valores de la opción de vinculación EXPLAIN			
	NO	**YES**	**REOPT**	**ALL**
RECOMMEND INDEXES	• Las tablas de explicación se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Las tablas de explicación se rellenan para SQL estático. • Las tablas de explicación se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Las tablas de explicación se rellenan para SQL estático. • Las tablas de explicación se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.
EVALUATE INDEXES	• Las tablas de explicación se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.	• Las tablas de explicación se rellenan para SQL estático. • Las tablas de explicación se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.	• Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de explicación se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.	• Las tablas de explicación se rellenan para SQL estático. • Las tablas de explicación se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.

El registro especial CURRENT EXPLAIN SNAPSHOT interactúa con la opción de vinculación EXPLSNAP de la siguiente manera para SQL dinámico.

Tabla 176. Interacción de la opción de vinculación EXPLSNAP y CURRENT EXPLAIN SNAPSHOT

Valores de EXPLAIN SNAPSHOT	Valores de la opción de vinculación EXPLSNAP			
	NO	**YES**	**REOPT**	**ALL**
NO	• Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático. • Se toma una Instantánea de explicación para SQL dinámico. • Se devuelven los resultados de la consulta.
YES	• Se toma una Instantánea de explicación para SQL dinámico. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático. • Se toma una Instantánea de explicación para SQL dinámico. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático. • Se toma una Instantánea de explicación para SQL dinámico. • Se devuelven los resultados de la consulta.

Tabla 176. Interacción de la opción de vinculación EXPLSNAP y CURRENT EXPLAIN SNAPSHOT (continuación)

Valores de EXPLAIN SNAPSHOT	Valores de la opción de vinculación EXPLSNAP			
	NO	YES	REOPT	ALL
EXPLAIN	• Se toma una Instantánea de explicación para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Se toma una Instantánea de explicación para SQL estático. • Se toma una Instantánea de explicación para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Se toma una Instantánea de explicación para SQL estático. • Se toma una Instantánea de explicación para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).
REOPT	• Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.

Apéndice K. Tablas de excepciones

Las tablas de excepciones son tablas creadas por el usuario que imitan la definición de las tablas cuya comprobación se especifica utilizando SET INTEGRITY con la opción IMMEDIATE CHECKED. Se utilizan para almacenar copias de las filas que violan las restricciones de las tablas que se están comprobando.

Las tablas de excepciones que se utilizan con LOAD son idénticas a las utilizadas aquí. Por lo tanto, se pueden volver a utilizar durante la comprobación con la sentencia SET INTEGRITY.

Reglas para crear una tabla de excepciones

Las reglas para crear una tabla de excepciones son las siguientes:

- Las primeras "n" columnas de la tabla de excepciones son iguales que las columnas de la tabla que se está comprobando. Todos los atributos de columna inclusive el nombre, el tipo y la longitud deben ser idénticos.
- Todas las columnas de la tabla de excepciones deben estar libres de cualquier restricción y activador. Las restricciones incluyen la integridad de referencia, las restricciones de comprobación así como las restricciones de índice de unicidad que podrían causar errores en la inserción.
- La columna "(n+1)" de la tabla de excepciones es una columna TIMESTAMP opcional. Esto sirve para identificar las invocaciones sucesivas de la comprobación que efectúa la sentencia SET INTEGRITY en la misma tabla, si las filas de la tabla de excepciones no se han suprimido antes mediante la emisión de la sentencia SET INTEGRITY para comprobar los datos.
- La columna "(n+2)" debe ser de tipo CLOB(32K) o mayor. Esta columna es opcional pero se recomienda incluirla y se utilizará para proporcionar los nombres de las restricciones que violan los datos de la fila. Si no se proporciona esta columna (como pasaría si, por ejemplo, la tabla original tuviese el número máximo de columnas permitido), sólo se copia la fila en la que se ha detectado la violación de restricción.
- La tabla de excepciones debe crearse con las columnas "(n+1)" y "(n+2)".
- No se impone ningún nombre en particular para las columnas adicionales anteriores. No obstante, debe seguirse exactamente la especificación del tipo.
- No se permiten columnas adicionales.
- Si la tabla original tiene columnas DATALINK, las columnas correspondientes de la tabla de excepciones deben especificar NO LINK CONTROL. Esto asegura que no se enlace un archivo cuando se inserte una fila (con columna DATALINK) ni se genere un símbolo de accesos cuando se seleccionen filas de la tabla de excepciones.
- Si la tabla original tiene columnas generadas (incluida la propiedad IDENTITY), las columnas correspondientes de la tabla de excepciones no deben especificar la propiedad generada.
- También debe tenerse en cuenta que los usuarios que invocan SET INTEGRITY para comprobar los datos deben tener el privilegio INSERT en las tablas de excepciones.

La información de la columna "mensaje" tendrá la siguiente estructura:

Reglas para crear una tabla de excepciones

Tabla 177. Estructura de la columna de mensajes de la tabla de excepciones

Número de campo	Contenido	Tamaño	Comentarios
1	Número de violaciones de restricción	5 caracteres	Justificada por la derecha rellenada con '0'
2	Tipo de la primera violación de restricción	1 carácter	'K' - Violación de restricción de comprobación 'F' - Violación de clave foránea 'G' - Violación de columna generada 'I' - Violación de índice de unicidad[a] 'L' - Violación de carga de DATALINK'D' - Violación de supresión de cascada
3	Longitud de restricción/columna [b] /ID índice[c]/DLVDESC [d]	5 caracteres	Justificada por la derecha rellenada con '0'
4	Nombre de restricción/Nombre de columna[b]/ID de índice[c]/DLVDESC[d]	longitud del campo anterior	
5	Separador	3 caracteres	<espacio><:><espacio>
6	Tipo de la siguiente violación de restricción	1 carácter	'K' - Violación de restricción de comprobación 'F' - Violación de clave foránea 'G' - Violación de columna generada 'I' - Violación de índice de unicidad 'L' - Violación de carga de DATALINK'D' - Violación de supresión de cascada
7	Longitud de restricción/columna/ID de índice/ DLVDESC	5 caracteres	Justificada por la derecha rellenada con '0'
8	Nombre de restricción/Nombre de columna/ID de índice/ DLVDESC	longitud del campo anterior	
.....	.....	.....	Repita del Campo 5 al 8 para cada violación

- [a] No se producirán violaciones de índices de unicidad en la comprobación si se utiliza SET INTEGRITY. Sin embargo, se informará de esto, cuando se ejecute LOAD si se elige la opción FOR EXCEPTION. Por otra parte, LOAD no informará de las violaciones de restricción de comprobación, de columna generada ni de clave foránea ocurridas en las tablas de excepciones.
- [b] Para recuperar la expresión de una columna generada a partir de las vistas de catálogo, utilice una sentencia de selección. Por ejemplo, si el campo 4 es MYSCHEMA.MYTABLE.GEN_1, entonces SELECT SUBSTR(TEXT, 1, 50) FROM SYSCAT.COLUMNS WHERE TABSCHEMA='MYSCHEMA' AND TABNAME='MYNAME' AND COLNAME='GEN_1'; devuelve los primeros 50 caracteres de la expresión, en el formato "AS (<expresión>)"
- [c] Para recuperar un ID de índice a partir de las vistas de catálogo, utilice una sentencia de selección. Por ejemplo, si el campo 4 es 1234, entonces SELECT INDSCHEMA, INDNAME FROM SYSCAT.INDEXES WHERE IID=1234.
- [d] DLVDESC es un DESCriptor de Violación de carga de DATALINK, que se describe a continuación.

Tabla 178. DESCriptor de Violación de carga de DATALINK (DLVDESC)

Número de campo	Contenido	Tamaño	Comentarios
1	Número de columnas DATALINK de violación	4 caracteres	Justificada por la derecha rellenada con '0'
2	Número de columna DATALINK de la primera columna de violación	4 caracteres	Justificada por la derecha rellenada con '0'
2	Número de columna DATALINK de la segunda columna de violación	4 caracteres	Justificada por la derecha rellenada con '0'
.....	.....	.....	Se repite para cada número de columna de violación

Nota:
- El número de columna DATALINK es COLNO en SYSCAT.COLUMNS para la tabla adecuada.

Gestión de las filas en una tabla de excepciones

La información de las tablas de excepciones se puede procesar de la forma que se desee. Las filas pueden utilizarse para corregir los datos y volver a insertar las filas en las tablas originales.

Si no hay ningún activador INSERT en la tabla original, transfiera las filas corregidas emitiendo la sentencia INSERT con una subconsulta en la tabla de excepciones.

Si hay activadores INSERT y desea completar la carga con las filas corregidas de las tablas de excepciones sin disparar los activadores, se sugieren las siguientes maneras:

- Diseñe los activadores INSERT para que se disparen dependiendo del valor de una columna definida explícitamente para esta finalidad.
- Descargue los datos de las tablas de excepciones y añádalos utilizando LOAD. En este caso, si se vuelven a comprobar los datos, se debe tener en cuenta que en DB2 Versión 8 la comprobación de la violación de restricciones no está confinada solamente a las filas añadidas.
- Guarde el texto del activador de la tabla de catálogos relevante. Después elimine el activador INSERT y utilice INSERT para transferir las filas corregidas de las tablas de excepciones. Finalmente vuelva a crear el activador utilizando la información guardada.

En DB2 Versión 8, no se realiza una provisión explícita para evitar que se disparen los activadores cuando se insertan filas desde las tablas de excepciones.

Sólo se informará de una violación por fila para las violaciones de índices de unicidad.

Si hay valores con series largas o con tipos de datos LOB en la tabla, los valores no se insertarán en la tabla de excepciones en caso de violación de índice de unicidad.

Consulta de las tablas de excepciones

La estructura de la columna de mensajes de una tabla de excepciones es una lista concatenada de nombres de restricciones, longitudes y delimitadores tal como se describe antes. Es posible que desee escribir una consulta sobre esta información.

Por ejemplo, supongamos que se escribe una consulta para obtener una lista de todas las violaciones, repitiendo cada fila con sólo el nombre de la restricción junto a ella. Supongamos que nuestra tabla original T1 tenía dos columnas C1 y C2. Supongamos también que la tabla de excepciones correspondiente E1 tiene las columnas C1, C2 que pertenecen a las de T1 y MSGCOL como la columna de mensajes. La siguiente consulta (utilizando recurrencia) listará un nombre de restricción por fila (que pertenece a la fila para más de una violación):

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, I, J) AS
  (SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, 12,
                   INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
       1,
       15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
     FROM E1
UNION ALL
   SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, J+6,
                  INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
       I+1,
       J9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
     FROM IV
     WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
 ) SELECT C1, C2, CONSTNAME FROM IV;
```

Si deseamos que todas las filas que han violado una restricción en particular, ampliaríamos esta consulta de la siguiente manera:

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, I, J) AS
  (SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, 12,
                   INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
       1,
       15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
     FROM E1
UNION ALL
   SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, J+6,
                  INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
       I+1,
       J9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
     FROM IV
     WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
 ) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTNAME = 'nombrerestricción';
```

Para obtener todas las violaciones de restricciones de comprobación, se podría ejecutar lo siguiente:

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, CONSTTYPE, I, J) AS
  (SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, 12,
                   INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
       CHAR(SUBSTR(MSGCOL, 6, 1)),
       1,
       15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
     FROM E1
UNION ALL
   SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, J+6,
```

```
                INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
      CHAR(SUBSTR(MSGCOL, J, 1)),
      I+1,
      J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
    FROM IV
    WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTTYPE = 'K';
```

Apéndice L. Sentencias de SQL que se permiten en rutinas

La tabla siguiente indica si se permite o no que la sentencia de SQL (especificada en la primera columna) se ejecute en una rutina que tenga especificada la indicación de acceso a datos SQL. Si se encuentra una sentencia de SQL ejecutable en una rutina definida con NO SQL, se devuelve SQLSTATE 38001. Para otros contextos de ejecución, las sentencias de SQL que no se soportan en ningún contexto devuelven SQLSTATE 38003. Para otras sentencias de SQL no permitidas en un contexto CONTAINS SQL, se devuelve SQLSTATE 38004. En un contexto READS SQL DATA, se devuelve SQLSTATE 38002. Durante la creación de una rutina SQL, una sentencia que no coincida con la indicación de acceso a datos SQL haría que se devolviera SQLSTATE 42985.

Si una sentencia invoca una rutina, la indicación de acceso a datos SQL efectiva para la sentencia será el mayor de:

- La indicación de acceso a datos SQL de la sentencia de la tabla siguiente.
- La indicación de acceso a datos SQL de la rutina especificada al crear la rutina.

Por ejemplo, la sentencia CALL tiene una indicación de acceso a datos SQL de CONTAINS SQL. Sin embargo, si se llama a un procedimiento almacenado definido como READS SQL DATA, la indicación de acceso a datos SQL efectiva para la sentencia CALL es READS SQL DATA.

Cuando una sentencia invoca una sentencia SQL, la indicación de acceso a datos SQL efectiva para la sentencia no debe sobrepasar la indicación de acceso a datos SQL declarada para la rutina. Por ejemplo, una función definida como READS SQL DATA no podría llamar a un procedimiento almacenado definido como MODIFIES SQL DATA.

Tabla 179. Sentencia de SQL e indicación del acceso a datos SQL

Sentencia de SQL	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
ALTER...	N	N	N	S
BEGIN DECLARE SECTION	S(1)	S	S	S
CALL	N	S	S	S
CLOSE	N	N	S	S
COMMENT ON	N	N	N	S
COMMIT	N	N(4)	N(4)	N(4)
COMPOUND SQL	N	S	S	S
CONNECT(2)	N	N	N	N
CREATE	N	N	N	S
DECLARE CURSOR	S(1)	S	S	S
DECLARE GLOBAL TEMPORARY TABLE	N	N	N	S
DELETE	N	N	N	S
DESCRIBE	N	S	S	S
DISCONNECT(2)	N	N	N	N

Tabla 179. Sentencia de SQL e indicación del acceso a datos SQL (continuación)

Sentencia de SQL	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
DROP ...	N	N	N	S
END DECLARE SECTION	S(1)	S	S	S
EXECUTE	N	S(3)	S(3)	S
EXECUTE IMMEDIATE	N	S(3)	S(3)	S
EXPLAIN	N	N	N	S
FETCH	N	N	S	S
FREE LOCATOR	N	S	S	S
FLUSH EVENT MONITOR	N	N	N	S
GRANT ...	N	N	N	S
INCLUDE	S(1)	S	S	S
INSERT	N	N	N	S
LOCK TABLE	N	S	S	S
OPEN	N	N	S (5)	S
PREPARE	N	S	S	S
REFRESH TABLE	N	N	N	S
RELEASE CONNECTION(2)	N	N	N	N
RELEASE SAVEPOINT	N	N	N	S
RENAME TABLE	N	N	N	S
REVOKE ...	N	N	N	S
ROLLBACK	N	N(4)	N(4)	N(4)
ROLLBACK TO SAVEPOINT	N	N	N	S
SAVEPOINT	N	N	N	S
SELECT INTO	N	N	S (5)	S
SET CONNECTION(2)	N	N	N	N
SET INTEGRITY	N	N	N	S
SET registro especial	N	S	S	S
UPDATE	N	N	N	S
VALUES INTO	N	N	S	S
WHENEVER	S(1)	S	S	S

Notas:

1. Aunque la opción NO SQL implica que no puede especificarse ninguna sentencia de SQL, las sentencias no ejecutables no están restringidas.
2. Las sentencias de gestión de conexiones no están permitidas en ningún contexto de ejecución de rutinas.
3. Depende de la sentencia que se ejecute. La sentencia especificada para la sentencia EXECUTE debe ser una sentencia que esté permitida en el contexto del nivel de acceso SQL que esté vigente. Por ejemplo, si el nivel acceso SQL en vigor es READS SQL DATA, la sentencia no deber ser INSERT, UPDATE ni DELETE.

4. Es posible utilizar la sentencia COMMIT y la sentencia ROLLBACK sin la cláusula TO SAVEPOINT en un procedimiento almacenado, pero sólo si se llama al procedimiento almacenado directamente desde una aplicación o indirectamente mediante llamadas a procedimientos almacenados anidados desde una aplicación. (Si alguna sentencia de activador, función, método o compuesto atómico está en la cadena de llamada al procedimiento almacenado, no se permite realizar COMMIT o ROLLBACK de una unidad de trabajo).
5. Si el nivel de acceso SQL en vigor es READS SQL DATA, no se puede incorporar ninguna sentencia de cambio de datos de SQL en la sentencia SELECT INTO ni el el cursor al que hace referencia la sentencia OPEN.

Apéndice M. CALL invocada desde una sentencia compilada

Invoca un procedimiento almacenado en la ubicación de una base de datos. Un procedimiento almacenado, por ejemplo, se ejecuta en la ubicación de la base de datos y devuelve los datos a la aplicación cliente.

Los programas que utilizan la sentencia de SQL CALL se diseñan para ejecutarse en dos partes, una en el cliente y la otra en el servidor. El procedimiento del servidor en la base de datos se ejecuta en la misma transacción que la aplicación cliente. Si la aplicación cliente y el procedimiento almacenado están en la misma partición, el procedimiento almacenado se ejecuta localmente.

Nota: Esta forma de la sentencia CALL está desestimada y sólo se suministra para la compatibilidad con las versiones anteriores de DB2.

Invocación:

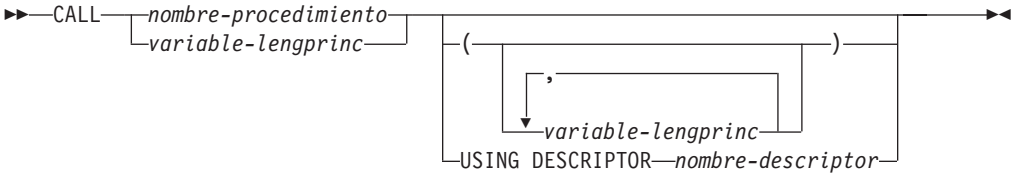
Esta forma de la sentencia CALL sólo puede incluirse en un programa de aplicación precompilado con la opción CALL_RESOLUTION DEFERRED. No puede utilizarse en activadores, procedimientos de SQL ni ningún otro contexto que no sea de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica. No obstante, el nombre de procedimiento puede especificarse mediante una variable del lenguaje principal y esto, junto con el uso de la cláusula USING DESCRIPTOR, permite proporcionar tanto el nombre de procedimiento como la lista de parámetros en tiempo de ejecución, con lo que se consigue un efecto similar al de una sentencia preparada dinámicamente.

Autorización:

Entre los privilegios que el ID de autorización de la sentencia CALL necesita poseer *durante la ejecución* debe incluirse por lo menos uno de los siguientes:

- Privilegio EXECUTE sobre el paquete asociado al procedimiento almacenado (el privilegio EXECUTE sobre el procedimiento almacenado *no* se comprueba.)
- Privilegio CONTROL sobre el paquete asociado al procedimiento almacenado
- Autorización SYSADM o DBADM

Sintaxis:



Descripción:

nombre-procedimiento **o** *variable-lengprinc*
Identifica el procedimiento que se va a llamar. El nombre de procedimiento puede especificarse directamente o dentro de una variable del lenguaje principal. El procedimiento identificado debe existir en el servidor actual (SQLSTATE 42724).

Si se especifica el *nombre-procedimiento*, debe ser un identificador normal que no sobrepase los 254 bytes. Como sólo puede ser un identificador normal, no puede contener blancos ni caracteres especiales. El valor se convierte a mayúsculas. Si es necesario utilizar nombres en minúsculas, blancos o caracteres especiales, el nombre debe especificarse mediante una *variable-lengprinc*.

Si se especifica *variable-lengprinc*, debe ser una variable serie-caracteres con un atributo de longitud que no sobrepase los 254 bytes y no debe incluir una variable indicadora. El valor *no* se convierte a mayúsculas. La serie de caracteres debe estar justificada por la izquierda.

El nombre de procedimiento puede tener uno de estos formatos.

nombre-procedimiento
El nombre (sin extensión) del procedimiento que se va a ejecutar. El procedimiento que se invoca se determina de la manera siguiente.

1. Se utiliza el *nombre-procedimiento* para buscar un procedimiento que coincida en los procedimientos definidos (en SYSCAT.ROUTINES). Un procedimiento que coincida se determina utilizando los pasos siguientes.
   a. Busque los procedimientos (ROUTINETYPE es 'P') del catálogo (SYSCAT.ROUTINES), donde ROUTINENAME coincida con el *nombre-procedimiento* especificado y ROUTINESCHEMA sea un nombre de esquema en la vía de acceso de SQL (registro especial CURRENT PATH). Si el nombre de esquema está especificado explícitamente, la vía de acceso de SQL se ignora y sólo se tienen en cuenta los procedimientos con el nombre de esquema especificado.
   b. Después, elimine cualquiera de estos procedimientos que no tengan el mismo número de parámetros que el número de argumentos especificados en la sentencia CALL.
   c. Elija el procedimiento restante que esté antes en la vía de acceso de SQL.

   Si se selecciona un procedimiento, DB2 invocará el procedimiento definido por el nombre externo.
2. Si no se encuentra ningún procedimiento que coincida, el se utiliza el *nombre-procedimiento* como el nombre de la biblioteca de procedimientos almacenados y el nombre de función dentro de dicha biblioteca. Por ejemplo, si el *nombre-procedimiento* es `proclib`, el servidor DB2 cargará la biblioteca de procedimientos almacenados `proclib` y ejecutará la rutina de función `proclib()` dentro de esa biblioteca.

   En los sistemas basados en UNIX, el directorio por omisión para las bibliotecas de procedimientos almacenados es `sqllib/function`. El directorio por omisión para los procedimientos almacenados no delimitados es`sqllib/function/unfenced`.

   En los sistemas basados en Windows, el directorio por omisión para las bibliotecas de procedimientos almacenados es `sqllib\function`. El directorio por omisión para los procedimientos almacenados no delimitados es`sqllib\function\unfenced`.

Si no se ha encontrado la biblioteca o función, se devuelve un error (SQLSTATE 42884).

biblioteca-procedimiento!nombre-función
El carácter de admiración (!) actúa como delimitador entre el nombre de biblioteca y el nombre de función del procedimiento almacenado. Por

ejemplo, si se especifica `proclib!func`, se carga `proclib` en memoria y se ejecuta la función `func` desde esa biblioteca. Esto permite que se coloquen múltiples funciones en la misma biblioteca de procedimientos almacenados.

La biblioteca de procedimientos almacenados está ubicada en los directorios o especificada en la variable LIBPATH, tal como se describe en el *nombre-procedimiento*.

vía-absoluta!nombre-función
La *vía-absoluta* especifica la vía de acceso completa a la biblioteca de procedimientos almacenados.

En un sistema basado en UNIX, por ejemplo, si se especifica `/u/terry/proclib!func`, se obtiene la biblioteca de procedimientos almacenados `proclib` del directorio `/u/terry` y se ejecuta la función `func` desde esa biblioteca.

En todos estos casos, la longitud total del nombre de procedimiento, incluida su vía de acceso completa implícita o explícita, no debe tener una longitud superior a 254 bytes.

(variable-lengprinc,...)
Cada especificación de *variable-lengprinc* es un parámetro de la sentencia CALL. El parámetro *n* de CALL corresponde al parámetro *n* del procedimiento almacenado del servidor.

Se supone que se utiliza cada *variable-lengprinc* para intercambiar datos en ambas direcciones entre cliente y servidor. Para evitar enviar datos innecesarios entre cliente y servidor, la aplicación cliente debe proporcionar una variable indicadora con cada parámetro y establecer el indicador en -1 si el parámetro no se utiliza para transmitir datos al procedimiento almacenado. El procedimiento almacenado debe establecer la variable indicadora en -128 para cualquier parámetro que no se utilice para devolver datos a la aplicación cliente.

Si el servidor es DB2 Universal Database, los parámetros deben tener tipos de datos coincidentes en el programa cliente y servidor.

USING DESCRIPTOR *nombre-descriptor*
Identifica una SQLDA que debe contener una descripción válida de las variables del lenguaje principal. El elemento SQLVAR *n* corresponde al parámetro *n* del procedimiento almacenado del servidor.

Antes de que se procese la sentencia CALL, la aplicación debe definir los campos siguientes de la SQLDA:

- SQLN para indicar el número de apariciones de SQLVAR proporcionadas en la SQLDA
- SQLDABC para indicar el número de bytes de almacenamiento asignados para la SQLDA
- SQLD para indicar el número de variables utilizadas en la SQLDA al procesar la sentencia
- Las apariciones de SQLVAR, para indicar los atributos de las variables. Deben inicializarse los siguientes campos de cada elemento pasado de SQLVAR base:
  - SQLTYPE
  - SQLLEN
  - SQLDATA

– SQLIND

Deben inicializarse los siguientes campos de cada elemento pasado de la SQLVAR secundaria:

– LEN.SQLLONGLEN
– SQLDATALEN
– SQLDATATYPE_NAME

Se supone que la SQLDA se utiliza para intercambiar datos en ambas direcciones entre cliente y servidor. Para evitar enviar datos innecesarios entre cliente y servidor, la aplicación cliente debe establecer el campo SQLIND en -1 si el parámetro no se utiliza para transmitir datos al procedimiento almacenado. El procedimiento almacenado debe establecer el campo SQLIND en -128 para cualquier parámetro que no se utilice para devolver datos a la aplicación cliente.

Notas:

- ***Utilización de tipos de datos de objeto grande (LOB)***:

  Si la aplicación cliente y servidora tiene que especificar datos LOB de una SQLDA, asigne el doble al número de entradas SQLVAR.

  Los procedimientos almacenados han soportado los tipos de datos LOB a partir de DB2 Versión 2. Estos tipos de datos no están soportados por todos los clientes o servidores de versiones anteriores.

- ***Recuperación del estado de retorno (RETURN_STATUS) resultante de un procedimiento SQL***:

  Si un procedimiento SQL emite satisfactoriamente una sentencia RETURN junto con un valor de estado, este valor se coloca en el primer campo SQLERRD de la SQLCA. Si la sentencia CALL se emite en un procedimiento SQL, utilice la sentencia GET DIAGNOSTICS para recuperar el valor de RETURN_STATUS. El valor es –1 cuando SQLSTATE indica un error.

- ***Conjuntos de resultados devueltos desde procedimientos almacenados***:

  Si el programa de aplicación cliente se escribe utilizando CLI, los conjuntos de resultados pueden devolverse directamente a la aplicación cliente. El procedimiento almacenado indica que un conjunto resultante debe devolverse declarando un cursor en ese conjunto resultante, abriendo un cursor en el conjunto resultante y dejando el cursor abierto al salir del procedimiento.

  Al final de un procedimiento:

  – Por cada cursor que se ha dejado abierto, se devuelve un conjunto resultante a la aplicación.
  – Si se deja abierto más de un cursor, los conjuntos del resultado se devuelven en el orden en que se han abierto sus cursores.
  – Sólo se devuelven las filas no leídas. Por ejemplo, si el conjunto resultante de un cursor tiene 500 filas y el procedimiento almacenado ha leído 150 de estas filas en el momento en que ha terminado el procedimiento almacenado, se devolverán las filas desde la 151 a la 500 al procedimiento almacenado.

- ***Manejo de registros especiales***:

  Los valores de los registros especiales utilizados para el llamador los hereda el procedimiento almacenado durante la invocación y se restauran al devolver el control al llamador. Los registros especiales se pueden modificar dentro de un procedimiento almacenado, pero estos cambios no tienen efecto en el llamador. Esto no es cierto para los procedimientos almacenados preexistentes (aquéllos definidos con el estilo de parámetro DB2DARI o situados en la biblioteca por

omisión), en los que los cambios hechos en los registros especiales de un procedimiento se convierten en los valores del llamador.

- *Compatibilidades*:

  Existe una forma nueva, preferida de la sentencia CALL que puede incluirse en una aplicación (precompilando la aplicación con la opción CALL_RESOLUTION IMMEDIATE) o que puede prepararse de forma dinámica.

Ejemplos:

Ejemplo 1:

En C, invoque un procedimiento denominado TEAMWINS en la biblioteca ACHIEVE, pasándole un parámetro almacenado en la variable del lenguaje principal HV_ARGUMENT.

```
strcpy(HV_PROCNAME, "ACHIEVE!TEAMWINS");
CALL :HV_PROCNAME (:HV_ARGUMENT);
```

Ejemplo 2:

En C, invoque un procedimiento denominado :SALARY_PROC, utilizando la SQLDA denominada INOUT_SQLDA.

```
struct sqlda *INOUT_SQLDA;
/* El código de configuración para variables SQLDA va aquí */
CALL :SALARY_PROC
USING DESCRIPTOR :*INOUT_SQLDA;
```

Ejemplo 3:

Un procedimiento almacenado Java se define en la base de datos, utilizando la sentencia siguiente:

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,
                                OUT COST DECIMAL(7,2),
                                OUT QUANTITY
INTEGER)
                  EXTERNAL NAME 'pieza!disponibles'
    LANGUAGE JAVA
    PARAMETER STYLE DB2GENERAL;
```

Una aplicación Java llama a este procedimiento almacenado utilizando el siguiente fragmento de código:

```
...
CallableStatement stpCall ;

 String sql = "CALL PARTS_ON_HAND (?,?,?)";

stpCall = con.prepareCall( sql ) ; /* con es la conexión */

stpCall.setInt( 1, variable1 ) ;
stpCall.setBigDecimal( 2, variable2 ) ;
stpCall.setInt( 3, variable3 ) ;

stpCall.registerOutParameter( 2, Types.DECIMAL, 2 ) ;
stpCall.registerOutParameter( 3, Types.INTEGER ) ;

stpCall.execute() ;

variable2 = stpCall.getBigDecimal(2) ;
variable3 = stpCall.getInt(3) ;
 ...
```

CALL invocada desde una sentencia compilada

Este fragmento de código de aplicación invocará el método Java *onhand* 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'.

Información relacionada:

- "Sentencia CALL" en la publicación *Consulta de SQL, Volumen 2*

Apéndice N. Consideraciones sobre el código UNIX ampliado (EUC) en japonés y chino tradicional

El Código UNIX ampliado (EUC) correspondiente a japonés y chino tradicional define un conjunto de reglas de codificación que pueden dar soporte a entre 1 y 4 juegos de caracteres. En algunos casos, como el de EUC japonés (eucJP) y EUC chino tradicional (eucTW), un carácter se puede codificar utilizando más d dos bytes. El uso de este tipo de esquema de codificación tiene implicaciones cuando se utiliza como la página de códigos del servidor de bases de datos o como el cliente de bases de datos. Las consideraciones clave incluyen las siguientes:

- Expansión o contracción de series cuando se realizan conversiones entre páginas de códigos EUC y páginas de códigos de doble byte
- El uso de Universal Character Set-2 (UCS-2) como la página de códigos para datos gráficos almacenados en un servidor de bases de datos definido con páginas de códigos eucJP (japonés) o eucTW (chino tradicional).

Con la excepción de estas consideraciones, el uso de EUC es coherente con el soporte del juego de caracteres de doble byte (DBCS). En este manual (y en otros), las referencias a *doble byte* se han modificado por *multi byte* para reflejar el soporte de reglas de codificación que permiten representaciones de caracteres que necesitan más de 2 bytes. Aquí se incluyen consideraciones detalladas sobre el soporte de EUC en japonés y chino tradicional. Esta información debe tenerla en cuenta cualquiera que utilice SQL con un servidor de bases de datos EUC o con un cliente de bases de datos EUC y cuando se utilice junto con información de desarrollo de aplicaciones.

Elementos de idioma

Caracteres

Cada carácter multi byte se considera una *letra* con la excepción del carácter blanco de doble byte, que se considera un *carácter especial*.

Señales

Las letras alfabéticas en minúsculas multi byte no se convierten a mayúsculas. Esto difiere de las letras alfabéticas minúsculas de un solo byte en señales, que generalmente se convierten a mayúsculas.

Identificadores

Identificadores de SQL

La conversión entre una página de códigos de doble byte y una página de códigos EUC puede dar lugar a la conversión de caracteres de doble byte en caracteres multi byte codificados con más de 2 bytes. Como resultado, un identificador que cumpla con la longitud máxima de la página de códigos de doble byte puede superar la longitud en la página de códigos EUC. La selección de identificadores para este tipo de entorno se debe realizar con cuidado para evitar la ampliación más allá de la longitud máxima del identificador.

Tipos de datos

Series de caracteres

En una base de datos MBCS, las series de caracteres pueden contener una combinación de caracteres de un juego de caracteres de un solo byte (SBCS) y de juegos de caracteres multi byte (MBCS). Cuando se utilizan estas series, las operaciones pueden proporcionar resultados diferentes si se basan en caracteres (los datos se tratan como caracteres) o si se basan en bytes (los datos se tratan como bytes). Compruebe la descripción de la función o de la operación para determinar cómo se procesan las series mixtas.

Series gráficas

Una serie gráfica es una secuencia de datos de caracteres de doble byte. Para permitir el almacenamiento de datos EUC en japonés o en chino tradicional en columnas gráficas, los caracteres EUC se codifican en UCS-2. Los caracteres que no son caracteres de doble byte bajo todos los esquemas de codificación soportados (por ejemplo, PC o EBCDIC DBCS) no se deben utilizar con columnas gráficas. Los resultados de utilizar caracteres que no sean los caracteres de doble byte puede dar lugar a la sustitución por caracteres de sustitución durante la conversión. La recuperación de dichos datos no devolverá el mismo valor que se ha entrado.

Asignaciones y comparaciones

Asignaciones de series: La conversión de una serie se realiza antes de la asignación. En los casos en los que interviene una página de códigos eucJP/eucTW y una página de códigos DBCS, una serie de caracteres se puede ampliar (DBCS a eucJP/eucTW) o reducir (eucJP/eucTW a DBCS). Esto puede dar lugar a errores en la asignación de almacenamiento y al truncado en la asignación de recuperación. Cuando el error en la asignación de almacenamiento se debe a la ampliación durante la conversión, se devuelve SQLSTATE 22524 en lugar de SQLSTATE 22001.

Paralelamente, las asignaciones en las que intervienen series gráficas pueden dar lugar a la conversión de un carácter de doble byte codificado UCS-2 en un carácter de sustitución en una página de códigos de PC o EBCDIC DBCS para caracteres que no tienen un carácter correspondiente de doble byte. Las asignaciones que sustituyen caracteres por caracteres de sustitución lo indicarán estableciendo el campo SQLWARN10 de la SQLCA en 'W'.

En casos de truncado durante la asignación de recuperación en la que intervienen series de caracteres multi byte, el punto de truncado puede formar parte de un carácter multi byte. En este caso, cada byte del fragmento de caracteres se sustituye por un blanco de un solo byte. Esto significa que pueden aparecer más de un blanco de un solo byte al final de una serie de caracteres truncada.

Comparaciones de series: Las comparaciones de series se realizan byte a byte. Las series de caracteres también utilizan la secuencia de clasificación definida para la base de datos. Las series gráficas no utilizan la secuencia de clasificación y, en una base de datos eucJP o eucTW, se codifican utilizando UCS-2. Por lo tanto, la comparación de dos series de caracteres mixtas puede dar lugar a un resultado diferente de la comparación de dos series gráficas, aunque contengan los mismos caracteres. Paralelamente, el orden de clasificación resultante de una columna de caracteres mixtos y una columna gráfica puede diferir.

Reglas para tipos de datos de resultados

El tipo de datos resultante para series de caracteres no se ve afectado por la posible ampliación de la serie. Por ejemplo, una unión de dos operandos CHAR

seguirá siendo un CHAR. Sin embargo, si uno de los operandos de la serie de caracteres se va a convertir de modo que la ampliación máxima haga que el atributo de longitud sea el más largo de los dos operandos, entonces el atributo de longitud de la serie de caracteres resultante se ve afectado. Por ejemplo, tengamos en cuenta las expresiones de resultado de una expresión CASE que tienen los tipos de datos VARCHAR(100) y VARCHAR(120). Supongamos que la expresión VARCHAR(100) es una variable del lenguaje principal de serie mixta (que puede necesitar conversión) y la expresión VARCHAR(120) es una columna en la base de datos eucJP. El tipo de datos resultante es VARCHAR(200), puesto que VARCHAR(100) se dobla para permitir la posible conversión. El mismo escenario sin la intervención de una base de datos eucJP o eucTW tendría el tipo de resultado VARCHAR(120).

Tenga en cuenta que el hecho de doblar la longitud de la variable del lenguaje principal se basa en el hecho de que el servidor de bases de datos es EUC en japonés o EUC en chino tradicional. Aunque el cliente sea también eucJP o eucTW, se aplica la operación de doblar. Esto permite que los clientes de doble byte y multi byte utilicen el mismo paquete de aplicaciones.

Reglas para conversiones de series

Los tipos de operaciones que se listan en la sección correspondiente del manual Consulta de SQL pueden convertir operandos a la página de códigos de la aplicación o de la base de datos.

Si estas operaciones se realizan en un entorno de páginas de códigos mixtas que incluya EUC en japonés o en chino tradicional, se puede producir la ampliación o contracción de operandos de series de caracteres mixtas. Por lo tanto, el tipo de datos resultando tiene un atributo de longitud que da cabida a la ampliación máxima, si es posible. En los casos en los que hay restricciones sobre el atributo de longitud de tipo de datos, se utiliza la longitud máxima permitida para el tipo de datos. Por ejemplo, en un entorno en el que el crecimiento máximo es el doble, una variable del lenguaje principal VARCHAR(200) se trata como si fuera VARCHAR(400), pero una variable del lenguaje principal CHAR(200) se trata como si fuera CHAR(254). Se puede producir un error en tiempo de ejecución cuando se realiza la conversión si la serie convertida supera la longitud máxima correspondiente al tipo de datos. Por ejemplo, la unión de CHAR(200) y CHAR(10) tendría el tipo de resultados CHAR(254). Cuando se convierte el valor de la parte izquierda de la operación UNION, se produce un error si se necesitan más de 254 caracteres.

En algunos casos, permitir el crecimiento máximo para la conversión hace que el atributo de longitud supere un límite. Por ejemplo, UNION sólo permite columnas de hasta 254 bytes. Por lo tanto, una consulta con una unión que incluyera una variable del lenguaje principal en la lista de columnas (a la que denominaremos :hv1) que fuera una serie de caracteres mixtos DBCS definida como una serie de caracteres de longitud variable de 128 bytes de longitud establecería el tipo de datos en VARCHAR(256), lo que daría lugar a un error al preparar la consulta, aunque parecería que la consulta en la aplicación no tuviera columnas de más de 254. En una situación en la que no sea probable que la serie real ocasione una ampliación más allá de 254 caracteres, se puede utilizar lo siguiente para preparar la sentencia.

```
SELECT CAST(:hv1 CONCAT ' AS VARCHAR(254)), C2 FROM T1
UNION
SELECT C1, C2 FROM T2
```

La concatenación de la serie nula con la variable del lenguaje principal forzará que la conversión se realice antes de que se ejecute la conversión. Esta consulta se puede preparar en el entorno DBCS a eucJP/eucTW, aunque puede producirse un error de truncado en el momento de la ejecución.

Esta técnica (concatenación de series nulas con conversión) se puede utilizar para manejar el límite de 254 bytes similar para SELECT DISTINCT o para utilizar la columna en cláusulas ORDER BY o GROUP BY.

Constantes

Constantes de series gráficas

El cliente EUC en japonés o chino tradicional puede contener caracteres de un solo byte o multi byte (como una serie de caracteres mixtos). La serie no debe contener más de 2.000 caracteres. Se recomienda utilizar en constantes gráficas únicamente caracteres que se conviertan en caracteres de doble byte en todas las páginas de códigos de doble byte de PC y EBCDIC relacionadas. Una constante de serie gráfica de una sentencia SQL se convierte de la página de códigos del cliente a la codificación de doble byte del servidor de bases de datos. Para un servidor EUC en japonés o chino tradicional, la constante se convierte a UCS-2, la codificación de doble byte utilizada para series gráficas. Para un servidor de doble byte, la constante se convierte de la página de códigos del cliente a la página de códigos DBCS del servidor.

Funciones

En el diseño de funciones definidas por el usuario se debe tener en cuenta el impacto del soporte de EUC en japonés y chino tradicional en los tipos de datos de parámetro. Una parte de la resolución de funciones tiene en cuenta los tipos de datos de los argumentos que se pasan a una llamada de función. Los argumentos de series de caracteres mixtos en los que interviene un cliente EUC en japonés o chino tradicional puede requerir bytes adicionales para especificar el argumento. Esto puede requerir que se modifique el tipo de datos para permitir un aumento de longitud. Por ejemplo, pueden necesitarse 4001 bytes para representar una serie de caracteres en la aplicación (un LONG VARCHAR) que quepa en una serie VARCHAR(4000) en el servidor. Si no se incluye una signatura de función que permita que el argumento sea LONG VARCHAR, la resolución de la función no encontrará una función.

Existen algunas funciones que no permiten series largas por varios motivos. El uso de argumentos LONG VARCHAR o CLOB con estas funciones no resultará satisfactorio. Por ejemplo, LONG VARCHAR como el segundo argumento de la función POSSTR integrada fallará en la resolución de función (SQLSTATE 42884).

Expresiones

Con el operador de concatenación

La ampliación potencial de uno de los operandos de la concatenación puede hacer que el tipo de datos y la longitud de los operandos concatenados se modifique cuando se trabaja en un entorno que incluya un servidor de ases de datos EUC en japonés o chino tradicional. Por ejemplo, con un servidor EUC en el que el valor de una variable del lenguaje principal puede doblar su longitud, tenga en cuenta el siguiente ejemplo.

```
CHAR200 CONCAT :char50
```

La columna *CHAR200* es de tipo CHAR(200). La variable del lenguaje principal *char50* está definida como CHAR(50). El tipo de resultado correspondiente a este operando de concatenación sería normalmente CHAR(250). Sin embargo, con un servidor de bases de datos eucJP o eucTW, la suposición es que la serie se puede ampliar hasta doblar la longitud. Por lo tanto, *char50* se trata como un CHAR(100) y el tipo de datos resultante es VARCHAR(300). Tenga en cuenta que, aunque el resultado sea un VARCHAR, siempre tendrá 300 bytes de datos, incluidos los blancos finales. Si no se desean los blancos finales adicionales, se debe definir la variable del lenguaje principal como VARCHAR(50) en lugar de CHAR(50).

Predicados

Predicado LIKE

Para un predicado LIKE en el que intervienen series de caracteres mixtos en una base de datos EUC:

- Un carácter de subrayado de media anchura SBCS hace referencia a un carácter SBCS.
- Un carácter de subrayado de anchura completa no SBCS hace referencia a un carácter no SBCS.
- Un carácter de signo de porcentaje de media anchura SBCS o de anchura completa no SBCS hace referencia a cero o más caracteres SBCS o no SBCS.

El carácter de escape debe ser un carácter SBCS o no SBCS. En una columna de caracteres, el carácter de escape también puede ser una serie binaria que contenga exactamente un byte.

Tenga en cuenta que el uso del carácter de subrayado puede producir diferentes resultados, en función de la página de códigos de la operación LIKE. Por ejemplo, los caracteres Katakana en EUC en japonés son caracteres multi byte (CS2), pero en la página de códigos DBCS en japonés son caracteres de un solo byte. Una consulta con el carácter de subrayado de un solo byte en la *expresión-patrón* devolvería ocurrencias de carácter Katakana en la posición del carácter de subrayado procedentes de un servidor DBCS en japonés. Sin embargo, las mismas filas procedentes de la tabla equivalente en un servidor EUC en japonés no se devolverían, puesto que los caracteres Katakana sólo coincidirán con un carácter de subrayado de doble byte.

Para un predicado LIKE en el que intervienen series gráficas en una base de datos EUC:

- Un carácter de subrayado de anchura completa (U+FF3F) hace referencia a un carácter Unicode.
- Un carácter de signo de porcentaje de anchura completa (U+FF05) hace referencia a cero o más caracteres Unicode.

Funciones

LENGTH

El proceso de esta función no difiere para series de caracteres mixtos en un entorno EUC. El valor devuelto es la longitud de la serie en la página de códigos del argumento. Al igual que en la Versión 8, si el argumento es una variable del lenguaje principal, el valor devuelto es la longitud de la serie en la página de códigos de la base de datos. Cuando se utiliza esta función para determinar la longitud de un valor, se debe tener en cuenta cómo se utiliza la longitud. Esto resulta especialmente cierto para constantes de series mixtas, ya que la longitud se

proporciona en bytes, no en caracteres. Por ejemplo, la longitud de una columna de series mixtas en una base de datos DBCS devuelta por la función LENGTH puede ser menor que la longitud del valor recuperado de dicha columna en un cliente eucJP o eucTW debido a la conversión de algunos caracteres DBCS en caracteres eucJP o eucTW multi byte.

SUBSTR

La función SUBSTR opera sobre series de caracteres mixtos byte a byte. La serie resultante puede incluir, por tanto, fragmentos de caracteres multi byte al principio o al final de la serie resultante. No se proporciona ningún proceso para detectar o procesar fragmentos de caracteres.

TRANSLATE

La función TRANSLATE da soporte a series de caracteres mixtos que incluyen caracteres multi byte. Los caracteres correspondientes de *exp-a-serie* y *exp-de-serie* deben tener el mismo número de bytes y no puede terminar con parte de un carácter multi byte.

La *exp-car-rell* debe dar lugar a un carácter de un solo byte cuando la *exp-serie-car* es una serie de caracteres. Puesto que TRANSLATE se ejecuta en la página de códigos de la *exp-serie-car*, la *exp-car-rell* se puede convertir de un carácter multi byte a un carácter de un solo byte.

En una *exp-serie-car* que termine con parte de un carácter multi byte no se convertirán estos bytes.

VARGRAPHIC

La función VARGRAPHIC sobre un operando de serie de caracteres en una página de códigos EUC en japonés o chino tradicional devuelve una serie gráfica en la página de códigos UCS-2.

- Los caracteres de un solo byte se convierten primero a su carácter correspondiente de doble byte en el juego de códigos al que pertenecen (eucJP o eucTW). Luego se convierten a la representación UCS-2 correspondiente. Si no hay ninguna representación de doble byte, el carácter se convierte al carácter de sustitución de doble byte definido para dicho juego de códigos antes de que se convierta a la representación UCS-2.
- Los caracteres de eucJP que son Katakana (eucJP CS2) son realmente caracteres de un solo byte en algunos esquemas de codificación. Por lo tanto, se convierten a los caracteres correspondientes de doble byte en eucJP o al carácter de sustitución de doble byte antes de que se conviertan a UCS-2.
- Los caracteres multi byte se convierten a sus representaciones UCS-2.

Sentencias

CONNECT

El proceso de una sentencia CONNECT satisfactoria devuelve información en la SQLCA que es importante cuando existen la posibilidad de que las aplicaciones procesen datos en un entorno que incluya una página de códigos EUC en japonés o chino tradicional en el cliente o servidor. El campo *SQLERRD(1)* proporciona la máxima ampliación de una serie de caracteres mixtos cuando se convierte de la página de códigos de la aplicación a la página de códigos de la base de datos. El campo *SQLERRD(2)* proporciona la máxima ampliación de una serie de caracteres mixtos cuando se convierte de la página de códigos de la base de datos a la página

de códigos de la aplicación. El valor es positivo si se puede producir una ampliación y negativo si se puede producir una contracción. Si el valor es negativo, el valor es siempre -1 ya que el peor de los casos que no se produzca contracción y que se necesite la longitud completa de la serie tras la conversión. El valor positivo máximo es 2, lo que significa que, en el peor de los casos, puede ser necesario doblar la longitud de la serie para la serie de caracteres tras la conversión.

La página de códigos del servidor de aplicaciones y del cliente de aplicaciones también están disponibles en el campo SQLERRMC de la SQLCA.

PREPARE

Los tipos de datos determinados para marcadores de parámetros sin tipo no se modifican en un entorno que incluye EUD en japonés o chino tradicional. Como resultado, puede ser necesario en algunos casos utilizar marcadores de parámetros con tipo para proporcionar longitud suficiente para series de caracteres mixtos en eucJP o eucTW. Por ejemplo, tenga en cuenta una inserción en una columna CHAR(10). La preparación de la sentencia:

```
INSERT INTO T1 (CH10) VALUES (?)
```

daría lugar a un tipo de datos CHAR(10) para el marcador de parámetro. Si el cliente fuera eucJP o eucTW, se necesitarían más de 10 bytes para representar la serie que se tiene que insertar, pero la misma serie en la página de códigos DBCS de la base de datos no tiene más de 10 bytes. En este caso, la sentencia para preparar incluiría un marcador de parámetro con tipo con una longitud mayor que 10. Por lo tanto, la preparación de la sentencia:

```
INSERT INTO T1 (CH10) VALUES (CAST(? AS VARCHAR(20))
```

daría lugar a un tipo de datos VARCHAR(20) para el marcador de parámetro.

Información relacionada:

- "Sentencia PREPARE" en la publicación *Consulta de SQL, Volumen 2*

Apéndice O. Especificaciones de formato de Backus-Naur (BNF) para los enlaces de datos

Un valor DATALINK es un valor encapsulado que contiene una referencia lógica de la base de datos a un archivo almacenado fuera de la base de datos.

El atributo de ubicación de datos de este valor encapsulado es una referencia lógica expresada en forma de un localizador uniforme de recursos (URL). El valor de este atributo sigue la sintaxis aplicable a los URL basada en el RFC 1738: Uniform Resource Locators (URL), T. Berners-Lee, L. Masinter, M. McCahill, December 1994 (BNF es un acrónimo de ″Backus-Naur Form″, una notación formal para describir la sintaxis de un lenguaje determinado.)

Se utilizan los convenios siguientes en la especificación BNF:

- "|" designa alternativas
- los corchetes [ ] delimitan elementos opcionales o repetidos
- los literales aparecen entre comillas dobles ""
- los elementos pueden ir precedidos por [*n*]* para representar *n* o más repeticiones del elemento que sigue; si no se especifica *n*, el valor por omisión es 0

Especificación BNF para enlaces de datos (DATALINK):

URL

```
url          =    httpurl | fileurl | uncurl | emptyurl
```

HTTP

```
httpurl      =    "http://" hostport [ "/" hpath ]
hpath        =    hsegment *[ "/" hsegment ]
hsegment     =    *[ uchar | ";" | ":" | "@" | "&" | "=" ]
```

Observe que se ha eliminado el elemento de búsqueda existente en la especificación BNF original definida en el RFC1738, pues no es una parte esencial de la referencia a archivo y no tiene ninguna utilidad en el contexto de los enlaces de datos.

FILE

```
fileurl      =    "file://" host "/" fpath
fpath        =    fsegment *[ "/" fsegment ]
fsegment     =    *[ uchar | "?" | ":" | "@" | "&" | "=" ]
```

Observe que `host` no es opcional y la serie ″`localhost`″ no tiene ningún significado especial, a diferencia de RFC1738. Con ello se evitan interpretaciones confusas de ″`localhost`″ en las configuraciones cliente/servidor y de bases de datos particionadas.

UNC

```
uncurl       =    "unc:\\" hostname "\" sharename "\" uncpath
sharename    =    *uchar
uncpath      =    fsegment *[ "\" fsegment ]
```

Se da soporte al convenio UNC habitual para nombres en Windows. Esto no es un modelo estándar en RFC1738.

EMPTYURL

```
emptyurl     =    ""
hostport     =    host [ ":" port ]
host         =    hostname | hostnumber
hostname     =    *[ domainlabel "." ] toplabel
domainlabel  =    alphadigit | alphadigit *[ alphadigit | "-" ] alphadigit
toplabel     =    alpha | alpha *[ alphadigit | "-" ] alphadigit
alphadigit   =    alpha | digit
hostnumber   =    digits "." digits "." digits "." digits
port         =    digits
```

Los URL vacíos (de longitud cero) también pueden utilizarse para valores de tipo DATALINK. Son útiles para actualizar columnas DATALINK cuando se notifican excepciones de reconciliación e intervienen columnas DATALINK que no pueden contener nulos. Se utiliza un URL de longitud cero para actualizar la columna y provocar la desconexión de un archivo.

Definiciones varias

```
lowalpha     =   "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
                 "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
                 "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
                 "y" | "z"
hialpha      =   "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
                 "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" |
                 "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" |
                 "Y" | "Z"
alpha        =   lowalpha | hialpha
digit        =   "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                 "8" | "9"
safe         =   "$" | "-" | "_" | "." | "+"
extra        =   "!" | "*" | "'" | "(" | ")" | ","
hex          =   digit | "A" | "B" | "C" | "D" | "E" | "F" |
                 "a" | "b" | "c" | "d" | "e" | "f"
escape       =   "%" hex hex
unreserved   =   alpha | digit | safe | extra
uchar        =   unreserved | escape
digits       =   1*digit
```

Los caracteres en blanco iniciales y de cola son eliminados por DB2 durante el análisis sintáctico. Además, los nombres de esquema ('HTTP', 'FILE', 'UNC') y de `lenguaje principal` no son sensibles a las mayúsculas/minúsculas y siempre se guardan en mayúsculas en la base de datos.

Apéndice P. Información técnica sobre DB2 Universal Database

Documentación y ayuda de DB2

Está disponible información técnica de DB2® a través de las herramientas y los métodos siguientes:

- Centro de información de DB2
  - Temas
  - Herramientas de ayuda para DB2
  - Programas de ejemplo
  - Guías de aprendizaje
- Archivos PDF descargables y en CD y manuales impresos
  - Guías
  - Manuales de consulta
- Ayuda de línea de mandatos
  - Ayuda de mandatos
  - Ayuda de mensajes
  - Ayuda para estados de SQL
- Código fuente instalado
  - Programas de ejemplo

Puede acceder a información técnica adicional de DB2 Universal Database™ como, por ejemplo, notas técnicas, white papers y Redbooks™ en línea en ibm.com®. Acceda al sitio de la biblioteca de software de gestión de información de DB2 en www.ibm.com/software/data/pubs/.

Actualizaciones de la documentación de DB2

De forma periódica, IBM® puede realizar FixPaks de la documentación y otras actualizaciones de la misma en el Centro de información de DB2 disponible. Si accede al Centro de información de DB2 en http://publib.boulder.ibm.com/infocenter/db2help/, siempre visualizará la información más actualizada. Si ha instalado el Centro de información de DB2 localmente, tendrá que instalar cualquier actualización de forma manual para poder visualizarla. Las actualizaciones de la documentación le permiten actualizar la información que ha instalado desde el *CD del Centro de información de DB2* cuando está disponible nueva información.

El Centro de información se actualiza con mayor frecuencia que los manuales PDF o en copia impresa. Para conseguir la información técnica de DB2 más actualizada, instale las actualizaciones de la documentación a medida que estén disponibles o diríjase al Centro de información de DB2 en el sitio www.ibm.com.

Conceptos relacionados:

- "CLI sample programs" en la publicación *CLI Guide and Reference, Volume 1*
- "Programas de ejemplo Java" en la publicación *Guía de desarrollo de aplicaciones: Creación y ejecución de aplicaciones*
- "Centro de información de DB2" en la página 812

Tareas relacionadas:

- "Invocación de ayuda según contexto desde una herramienta de DB2" en la página 830
- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 822
- "Invocación de la ayuda de mensajes desde el procesador de línea de mandatos" en la página 832
- "Invocación de la ayuda de mandatos desde el procesador de línea de mandatos" en la página 832
- "Invocación de la ayuda para estados de SQL desde el procesador de línea de mandatos" en la página 833

Información relacionada:

- "Documentación PDF e impresa de DB2" en la página 824

Centro de información de DB2

El Centro de información de DB2® le proporciona acceso a toda la información que necesita para obtener el máximo provecho de los productos de la familia de DB2, incluidos DB2 Universal Database™, DB2 Connect™, DB2 Information Integrator y DB2 Query Patroller™. El Centro de información de DB2 también contiene información relativa a las características y los componentes principales de DB2, como la duplicación, el depósito de datos y DB2 Extenders.

El Centro de información de DB2 presenta las características siguientes si se visualiza en Mozilla 1.0 o posterior o bien en Microsoft® Internet Explorer 5.5 o posterior. Algunas características requieren que se habilite el soporte de JavaScript™:

Opciones flexibles de instalación
Puede elegir visualizar la documentación de DB2 utilizando la opción que mejor se ajuste a sus necesidades:

- Para asegurarse fácilmente de que la documentación siempre esté actualizada, puede acceder a toda la documentación directamente desde el Centro de información de DB2 incluido en el sitio Web de IBM® de http://publib.boulder.ibm.com/infocenter/db2help/
- Para minimizar el esfuerzo de actualización y mantener el tráfico de red en su intranet, puede instalar la documentación de DB2 en un solo servidor de la intranet
- Para maximizar la flexibilidad y reducir la dependencia de las conexiones de red, puede instalar la documentación de DB2 en su propio sistema

Búsqueda
Es posible buscar en todos los temas del Centro de información de DB2 entrando un término de búsqueda en el campo de texto **Buscar**. Puede recuperar coincidencias exactas encerrando los términos entre comillas y puede afinar la búsqueda mediante operadores de comodín (*, ?) y operadores booleanos (AND, NOT, OR).

Tabla de contenido orientada a tareas
Puede localizar los temas en la documentación de DB2 a partir de una sola tabla de contenido. La tabla de contenido está organizada principalmente

según la clase de tareas que puede desear realizar, pero también incluye entradas para visiones generales de productos, objetivos, información de consulta, un índice y un glosario.

- Las visiones generales de los productos describen la relación entre los productos disponibles en la familia de DB2, las características que ofrece cada uno de estos productos y proporcionan información actualizada del release de cada uno de estos productos.
- Las categorías de objetivos, como la instalación, la administración y el desarrollo, incluyen temas que permiten realizar rápidamente tareas y desarrollar un conocimiento más profundo de la información de fondo para realizar dichas tareas.
- Los temas de consulta proporcionan información detallada sobre un tema, incluida la sintaxis de sentencias y mandatos, la ayuda de mensajes y los parámetros de configuración.

Mostrar el tema actual en la tabla de contenido
Puede mostrar dónde encaja el tema actual en la tabla de contenido pulsando el botón **Renovar / Mostrar tema actual** en el marco de la tabla de contenido o pulsando el botón **Mostrar en tabla de contenido** en el marco del contenido. Esta característica es útil si ha seguido varios enlaces con temas relacionados en varios archivos o ha llegado a un tema a partir de resultados de una búsqueda.

Índice Es posible acceder a toda la documentación desde el índice. El índice está organizado en orden alfabético por términos del índice.

Glosario
Puede utilizar el glosario a fin de buscar definiciones de términos utilizados en la documentación de DB2. El glosario está organizado en orden alfabético por términos del glosario.

Información adaptada integrada
El Centro de información de DB2 visualiza la información en el idioma preferido que se ha establecido en las preferencias de navegador. Si un tema no está disponible en el idioma preferido del usuario, el Centro de información de DB2 visualiza la versión inglesa de ese tema.

Si desea información técnica sobre iSeries™, consulte el centro de información de IBM eServer™ iSeries en www.ibm.com/eserver/iseries/infocenter/.

Conceptos relacionados:

- "Escenarios de instalación del Centro de información de DB2" en la página 814

Tareas relacionadas:

- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 822
- "Visualización de temas en el idioma preferido en el Centro de información de DB2" en la página 823
- "Invocación del Centro de información de DB2" en la página 821
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (UNIX)" en la página 816
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (Windows)" en la página 819

Escenarios de instalación del Centro de información de DB2

Los entornos de trabajo distintos pueden plantear requisitos distintos para el modo de acceder a la información de DB2®. Se puede acceder al Centro de información de DB2 en el sitio Web de IBM®, en un servidor de la red de la organización o en una versión instalada en el sistema. En los tres casos, la documentación está incluida en el Centro de información de DB2, el cual consiste en una Web estructurada de información que se organiza en temas y que se visualiza mediante un navegador. Por omisión, los productos de DB2 acceden al Centro de información de DB2 en el sitio Web de IBM. No obstante, si desea acceder al Centro de información de DB2 en un servidor de intranet o en su propio sistema, es necesario que instale el Centro de información de DB2 utilizando el CD del Centro de información de DB2 que encontrará en el Paquete de soportes del producto. Consulte el siguiente resumen de opciones para acceder a la documentación de DB2, junto con los tres escenarios de instalación, como ayuda para determinar qué método de acceso al Centro de información de DB2 le funciona mejor en su entorno de trabajo y qué cuestiones relacionadas con la instalación se pueden tener en cuenta.

Resumen de opciones para acceder a la documentación de DB2:

La siguiente tabla proporciona recomendaciones sobre las opciones que son posibles en su entorno de trabajo a la hora de acceder a la documentación de productos de DB2 del Centro de información de DB2.

Acceso a Internet	Acceso a Intranet	Recomendación
Sí	Sí	Acceda al Centro de información de DB2 en el sitio Web de IBM o acceda al Centro de información de DB2 instalado en un servidor de intranet.
Sí	No	Acceda al Centro de información de DB2 en el sitio Web de IBM.
No	Sí	Acceda al Centro de información de DB2 instalado en un servidor de intranet.
No	No	Acceda al Centro de información de DB2 en un sistema local.

Escenario: Acceso al Centro de información de DB2 en su sistema:

Tsu-Chen es propietario de una fábrica en una pequeña ciudad que no dispone de ISP local para proporcionarle acceso a Internet. Ha adquirido DB2 Universal Database™ para la gestión de su inventario, pedidos de productos, información de cuentas bancarias y gastos empresariales. Puesto que nunca había utilizado un producto de DB2 anteriormente, Tsu-Chen tendrá que aprender a partir de la documentación de productos de DB2.

Después de instalar DB2 Universal Database en el sistema utilizando la opción de instalación típica, Tsu-Chen intenta acceder a la documentación de DB2. Sin embargo, el navegador emite un mensaje de error que indica que la página que ha intentado abrir no se encuentra. Tsu-Chen comprueba el manual de instalación de su producto de DB2 y descubre que tiene que instalar el Centro de información de DB2 si desea acceder a la documentación de DB2 en su sistema. Encuentra el *CD del Centro de información de DB2* en el paquete de soportes y lo instala.

Desde el programa ejecutor de aplicaciones del sistema operativo, Tsu-Chen dispone ahora de acceso al Centro de información de DB2 y puede aprender a utilizar el producto de DB2 para incrementar el éxito de su empresa.

Escenario: Acceso al Centro de información de DB2 en el sitio Web de IBM:

Colin es un consultor de tecnologías de la información con una empresa de formación. Está especializado en tecnología de bases de datos y SQL y ofrece clases sobre estos temas a empresas por toda Norteamérica utilizando DB2 Universal Database. Parte de las clases de Colin incluye el uso de la documentación de DB2 como una herramienta didáctica. Por ejemplo, mientras imparte los cursos sobre SQL, Colin utiliza la documentación de DB2 relativa a SQL como un modo de enseñar sintaxis básica y avanzada para las consultas de base de datos.

La mayoría de las empresas en las que Colin imparte cursos tienen acceso a Internet. Esta situación ha influido en la decisión de Colin de configurar su sistema portátil para que acceda al Centro de información de DB2 en el sitio Web de IBM cuando ha instalado la versión más reciente de DB2 Universal Database. Dicha configuración permite a Colin disponer de acceso en línea a la documentación más reciente de DB2 durante sus clases.

Sin embargo, a veces, mientras viaja, Colin no tiene acceso a Internet. Esto le planteaba un problema, especialmente cuando necesitaba acceder a la documentación de DB2 para preparar las clases. A fin de evitar tales situaciones, Colin ha instalado una copia del Centro de información de DB2 en el sistema portátil.

Colin disfruta de la flexibilidad que supone tener siempre una copia de la documentación de DB2 a su disposición. Mediante el mandato **db2set**, puede configurar fácilmente las variables de registro en el sistema portátil para acceder al Centro de información de DB2 en el sitio Web de IBM o en el sistema portátil, según su situación.

Escenario: Acceso al Centro de información de DB2 en un servidor de intranet:

El trabajo de Eva es el de administrador sénior de bases de datos en una compañía de seguros de vida. Sus responsabilidades administrativas incluyen la instalación y configuración de la versión más reciente de DB2 Universal Database en los servidores de bases de datos UNIX® de la compañía. Recientemente, la compañía ha informado a sus empleados de que, por razones de seguridad, no se les proporcionará acceso a Internet en el trabajo. Dado que la compañía tiene un entorno de red, Eva decide instalar una copia del Centro de información de DB2 en un servidor de intranet a fin de que todos los empleados de la compañía que utilicen el depósito de datos de la misma de forma regular (representantes de ventas, gestores de ventas y analistas de empresa) tengan acceso a la documentación de DB2.

Eva indica a su equipo encargado de las bases de datos que instalen la versión más reciente de DB2 Universal Database en los sistemas de todos los empleados a través de un archivo de respuestas, para asegurarse de que cada sistema esté configurado de manera que acceda al Centro de información de DB2 utilizando el nombre de sistema principal y el número de puerto del servidor de intranet.

No obstante, debido a un malentendido, Miguel, un administrador de bases de datos auxiliar del equipo de Eva, instala una copia del Centro de información de DB2 en varios sistemas de los empleados en lugar de configurar DB2 Universal

Database para que acceda al Centro de información de DB2 en el servidor de intranet. Con el fin de corregir esta situación, Eva indica a Miguel que utilice el mandato **db2set** para cambiar las variables de registro del Centro de información de DB2 (DB2_DOCHOST para el nombre de sistema principal y DB2_DOCPORT para el número de puerto) en cada uno de esos sistemas. Ahora todos los sistemas correspondientes de la red tienen acceso al Centro de información de DB2, y los empleados pueden hallar las respuestas a sus preguntas sobre DB2 en la documentación de DB2.

Conceptos relacionados:

- "Centro de información de DB2" en la página 812

Tareas relacionadas:

- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 822
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (UNIX)" en la página 816
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (Windows)" en la página 819
- "Establecimiento de la ubicación para acceder al Centro de información de DB2"

Información relacionada:

- "db2set - Mandato Registro de perfiles de DB2" en la publicación *Consulta de mandatos*

Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (UNIX)

Se puede acceder a la documentación de los productos de DB2 de tres maneras: en el sitio Web de IBM, en un servidor de intranet o en una versión instalada en el sistema. Por omisión, el acceso de los productos de DB2 dentro de la documentación de DB2 se efectúa en el sitio Web de IBM. Si desea acceder a la documentación de DB2 en un servidor de intranet o en su propio sistema, deberá instalar la documentación desde el *CD del Centro de información de DB2*. Mediante el asistente de instalación de DB2, puede definir sus preferencias de instalación e instalar el Centro de información de DB2 en un sistema que utilice un sistema operativo UNIX.

Prerrequisitos:

Este apartado lista los requisitos de hardware, sistema operativo, software y comunicaciones para instalar el Centro de información de DB2 en los sistemas UNIX.

- **Requisitos de hardware**

  Necesita uno de los procesadores siguientes:

  – PowerPC (AIX)
  – HP 9000 (HP-UX)
  – Intel de 32 bits (Linux)
  – Sistemas Solaris UltraSPARC (Entorno operativo Solaris)
- **Requisitos de sistema operativo**

  Necesita uno de los sistemas operativos siguientes:

- IBM AIX 5.1 (en PowerPC)
- HP-UX 11i (en HP 9000)
- Red Hat Linux 8.0 (en Intel de 32 bits)
- SuSE Linux 8.1 (en Intel de 32 bits)
- Sun Solaris Versión 8 (en sistemas UltraSPARC del Entorno operativo Solaris)

Nota: El Centro de información de DB2 se ejecuta en un subconjunto de los sistemas operativos UNIX en los que están soportados los clientes DB2. Por consiguiente, es recomendable que acceda al Centro de información de DB2 desde el sitio Web de IBM o que instale el Centro de información de DB2 y acceda al mismo en un servidor de intranet.

- **Requisitos de software**
  - Está soportado el navegador siguiente:
    - Mozilla Versión 1.0 o superior
- El asistente de instalación de DB2 es un instalador gráfico. Debe disponer de una implementación del software X Window System capaz de representar una interfaz gráfica de usuario para que el asistente de instalación de DB2 se ejecute en el sistema. A fin de ejecutar el asistente de instalación de DB2, debe asegurarse de que ha exportado debidamente la visualización. Por ejemplo, entre el mandato siguiente en el indicador de mandatos:

  ```
  export DISPLAY=9.26.163.144:0.
  ```

- **Requisitos de comunicaciones**
  - TCP/IP

Procedimiento:

Para instalar el Centro de información de DB2 utilizando el asistente de instalación de DB2:

1. Inicie una sesión en el sistema.
2. Inserte y monte el CD del producto Centro de información de DB2 en el sistema.
3. Vaya al directorio en el que está montado el CD entrando el mandato siguiente:

   ```
   cd /cd
   ```

   donde */cd* representa el punto de montaje del CD.
4. Entre el mandato **./db2setup** para iniciar el asistente de instalación de DB2.
5. Se abrirá el Área de ejecución para la instalación de IBM DB2. Para continuar directamente con la instalación del Centro de información de DB2, pulse en **Instalar producto**. Existe ayuda en línea disponible para guiarle durante los pasos restantes. Para invocar la ayuda en línea, pulse en **Ayuda**. Puede pulsar en **Cancelar** en cualquier momento para interrumpir la instalación.
6. En la página **Seleccione el producto que desee instalar**, pulse en **Siguiente**.
7. Pulse en **Siguiente** en la página **Bienvenido al asistente de instalación de DB2**. El asistente de instalación de DB2 le guiará durante el proceso de instalación del programa.
8. Para continuar con la instalación, debe aceptar el contrato de licencia. En la página **Contrato de licencia**, seleccione **Acepto los términos del contrato de licencia** y pulse en **Siguiente**.
9. Seleccione **Instalar el Centro de información de DB2 en este sistema** en la página **Seleccionar la acción de instalación**. Si desea utilizar un archivo de

respuestas para instalar el Centro de información de DB2 en éste o en otros sistemas más adelante, seleccione **Guardar los valores en un archivo de respuestas**. Pulse en **Siguiente**.

10. Seleccione los idiomas en los que se instalará el Centro de información de DB2 en la página **Seleccionar los idiomas a instalar**. Pulse en **Siguiente**.
11. Configure el Centro de información de DB2 para las comunicaciones entrantes en la página **Especificar el puerto del Centro de información de DB2**. Pulse en **Siguiente** para continuar la instalación.
12. Revise las opciones de instalación que ha elegido en la página **Comenzar a copiar archivos**. Para cambiar cualquier valor, pulse en **Anterior**. Pulse en **Instalar** para copiar los archivos del Centro de información de DB2 en el sistema.

También puede instalar el Centro de información de DB2 utilizando un archivo de respuestas.

Los archivos de anotaciones cronológicas de instalación db2setup.his, db2setup.log y db2setup.err están ubicados, por omisión, en el directorio /tmp.

El archivo db2setup.log capta toda la información de instalación del producto de DB2, incluidos los errores. El archivo db2setup.his registra todas las instalaciones de productos de DB2 en el sistema. DB2 añade el archivo db2setup.log al archivo db2setup.his. El archivo db2setup.err capta cualquier salida de errores devuelta por Java, como, por ejemplo, información de interrupciones y excepciones.

Cuando se haya completado la instalación, el Centro de información de DB2 estará instalado en uno de los directorios siguientes, según el sistema operativo UNIX:

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Entorno operativo Solaris: /opt/IBM/db2/V8.1

Conceptos relacionados:

- "Centro de información de DB2" en la página 812
- "Escenarios de instalación del Centro de información de DB2" en la página 814

Tareas relacionadas:

- "Instalación de DB2 utilizando un archivo de respuestas (UNIX)" en la publicación *Suplemento de instalación y configuración*
- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 822
- "Visualización de temas en el idioma preferido en el Centro de información de DB2" en la página 823
- "Invocación del Centro de información de DB2" en la página 821
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (Windows)" en la página 819

Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (Windows)

Se puede acceder a la documentación de los productos de DB2 de tres maneras: en el sitio Web de IBM, en un servidor de intranet o en una versión instalada en el sistema. Por omisión, el acceso de los productos de DB2 dentro de la documentación de DB2 se efectúa en el sitio Web de IBM. Si desea acceder a la documentación de DB2 en un servidor de intranet o en su propio sistema, deberá instalar la documentación de DB2 desde el *CD del Centro de información de DB2*. Mediante el asistente de instalación de DB2, puede definir sus preferencias de instalación e instalar el Centro de información de DB2 en un sistema que utilice un sistema operativo Windows.

Prerrequisitos:

Este apartado lista los requisitos de hardware, sistema operativo, software y comunicaciones para instalar el Centro de información de DB2 en Windows.

- **Requisitos de hardware**

  Necesita uno de los procesadores siguientes:

  – Sistemas de 32 bits: una CPU Pentium o compatible con Pentium
- **Requisitos de sistema operativo**

  Necesita uno de los sistemas operativos siguientes:

  – Windows 2000

  – Windows XP

  Nota: El Centro de información de DB2 se ejecuta en un subconjunto de los sistemas operativos Windows en los que están soportados los clientes DB2. Por consiguiente, es recomendable que acceda al Centro de información de DB2 en el sitio Web de IBM o que instale el Centro de información de DB2 y acceda al mismo en un servidor de intranet.
- **Requisitos de software**

  – Están soportados los navegadores siguientes:

    - Mozilla 1.0 o superior

    - Internet Explorer Versión 5.5 ó 6.0 (Versión 6.0 para Windows XP)
- **Requisitos de comunicaciones**

  – TCP/IP

Restricciones:

- Necesita una cuenta con privilegios administrativos para instalar el Centro de información de DB2.

Procedimiento:

Para instalar el Centro de información de DB2 utilizando el asistente de instalación de DB2:

1. Inicie una sesión en el sistema con la cuenta que ha definido para la instalación del Centro de información de DB2.
2. Inserte el CD en la unidad. Si está habilitada, la característica de ejecución automática inicia el Área de ejecución para la instalación de IBM DB2.
3. El asistente de instalación de DB2 determina el idioma del sistema y ejecuta el programa de instalación para ese idioma. Si desea ejecutar el programa de

instalación en un idioma distinto del inglés o bien el programa de instalación no se inicia de forma automática, puede iniciar el asistente de instalación de DB2 manualmente.

Para iniciar el asistente de instalación de DB2 manualmente:

a. Pulse en **Inicio** y seleccione **Ejecutar**.
b. En el campo **Abrir**, escriba el mandato siguiente:

```
x:\setup.exe /i identificador de idioma de 2 letras
```

donde *x:* representa la unidad de CD, e *identificador de idioma de 2 letras* representa el idioma en el que se ejecutará el programa de instalación.

c. Pulse en **Aceptar**.

4. Se abrirá el Área de ejecución para la instalación de IBM DB2. Para continuar directamente con la instalación del Centro de información de DB2, pulse en **Instalar producto**. Existe ayuda en línea disponible para guiarle durante los pasos restantes. Para invocar la ayuda en línea, pulse en **Ayuda**. Puede pulsar en **Cancelar** en cualquier momento para interrumpir la instalación.
5. En la página **Seleccione el producto que desee instalar**, pulse en **Siguiente**.
6. Pulse en **Siguiente** en la página **Bienvenido al asistente de instalación de DB2**. El asistente de instalación de DB2 le guiará durante el proceso de instalación del programa.
7. Para continuar con la instalación, debe aceptar el contrato de licencia. En la página **Contrato de licencia**, seleccione **Acepto los términos del contrato de licencia** y pulse en **Siguiente**.
8. Seleccione **Instalar el Centro de información de DB2 en este sistema** en la página **Seleccionar la acción de instalación**. Si desea utilizar un archivo de respuestas para instalar el Centro de información de DB2 en éste o en otros sistemas más adelante, seleccione **Guardar los valores en un archivo de respuestas**. Pulse en **Siguiente**.
9. Seleccione los idiomas en los que se instalará el Centro de información de DB2 en la página **Seleccionar los idiomas a instalar**. Pulse en **Siguiente**.
10. Configure el Centro de información de DB2 para las comunicaciones entrantes en la página **Especificar el puerto del Centro de información de DB2**. Pulse en **Siguiente** para continuar la instalación.
11. Revise las opciones de instalación que ha elegido en la página **Comenzar a copiar archivos**. Para cambiar cualquier valor, pulse en **Anterior**. Pulse en **Instalar** para copiar los archivos del Centro de información de DB2 en el sistema.

Puede instalar el Centro de información de DB2 utilizando un archivo de respuestas. También es posible utilizar el mandato **db2rspgn** a fin de generar un archivo de respuestas basado en una instalación existente.

Para obtener información sobre los errores encontrados durante la instalación, consulte los archivos db2.log y db2wi.log ubicados en el directorio 'Mis documentos'\DB2LOG\. La ubicación del directorio 'Mis documentos' dependerá de la configuración de su sistema.

El archivo db2wi.log capta la información de la instalación de DB2 más reciente. El archivo db2.log capta el historial de instalaciones de productos de DB2.

Conceptos relacionados:

- "Centro de información de DB2" en la página 812

- "Escenarios de instalación del Centro de información de DB2" en la página 814

Tareas relacionadas:

- "Instalación de un producto DB2 utilizando un archivo de respuestas (Windows)" en la publicación *Suplemento de instalación y configuración*
- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 822
- "Visualización de temas en el idioma preferido en el Centro de información de DB2" en la página 823
- "Invocación del Centro de información de DB2" en la página 821
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (UNIX)" en la página 816

Información relacionada:

- "db2rspgn - Mandato del Generador de archivos de respuestas (Windows)" en la publicación *Consulta de mandatos*

Invocación del Centro de información de DB2

El Centro de información de DB2 proporciona acceso a toda la información que necesita para utilizar productos de DB2 para los sistemas operativos Linux, UNIX y Windows, tales como DB2 Universal Database, DB2 Connect, DB2 Information Integrator y DB2 Query Patroller.

Puede invocar el Centro de información de DB2 desde una de las ubicaciones siguientes:

- Sistemas en los que está instalado un cliente o servidor DB2 UDB
- Un servidor de intranet o sistema local en el que está instalado el Centro de información de DB2
- El sitio Web de IBM

Prerrequisitos:

Antes de invocar el Centro de información de DB2:

- *Opcional*: Configure el navegador para que visualice los temas en su idioma preferido
- *Opcional*: Configure el cliente DB2 para que utilice el Centro de información de DB2 instalado en el sistema o servidor de intranet

Procedimiento:

Para invocar el Centro de información de DB2 en un sistema en el que está instalado un cliente o servidor DB2 UDB:

- Desde el menú Inicio (sistema operativo Windows): Pulse en **Inicio** → **Programas** → **IBM DB2** → **Información** → **Centro de información**.
- Desde el indicador de línea de mandatos:
  - En los sistemas operativos Linux y UNIX, emita el mandato **db2icdocs**.
  - En el sistema operativo Windows, emita el mandato **db2icdocs.exe**.

Para abrir el Centro de información de DB2 instalado en un servidor de intranet o sistema local en un navegador Web:

- Abra la página Web en http://<nombre-sistemaprincipal>:<número-puerto>/, donde <nombre-sistemaprincipal> representa el nombre de sistema principal y <número-puerto> representa el número de puerto en el que está disponible el Centro de información de DB2.

Para abrir el Centro de información de DB2 en el sitio Web de IBM en un navegador Web:

- Abra la página Web en publib.boulder.ibm.com/infocenter/db2help/.

Conceptos relacionados:

- "Centro de información de DB2" en la página 812
- "Escenarios de instalación del Centro de información de DB2" en la página 814

Tareas relacionadas:

- "Visualización de temas en el idioma preferido en el Centro de información de DB2" en la página 823
- "Invocación de ayuda según contexto desde una herramienta de DB2" en la página 830
- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 822
- "Invocación de la ayuda de mandatos desde el procesador de línea de mandatos" en la página 832
- "Establecimiento de la ubicación para acceder al Centro de información de DB2"

Información relacionada:

- "Mandato HELP" en la publicación *Consulta de mandatos*

Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet

El Centro de información de DB2 que hay disponible en http://publib.boulder.ibm.com/infocenter/db2help/ se actualizará periódicamente con documentación nueva o modificada. Asimismo, IBM puede efectuar actualizaciones del Centro de información de DB2 disponibles para descargar e instalar en el sistema o servidor de intranet. La actualización del Centro de información de DB2 no actualiza los productos de cliente o servidor DB2.

Prerrequisitos:

Es necesario tener acceso a un sistema que esté conectado a Internet.

Procedimiento:

Para actualizar el Centro de información de DB2 instalado en el sistema o servidor de intranet:

1. Abra el Centro de información de DB2 que se encuentra en el sitio Web de IBM de: http://publib.boulder.ibm.com/infocenter/db2help/
2. En la sección de descargas de la página de bienvenida, bajo la cabecera de servicio y soporte, pulse en el enlace de **documentación de DB2 Universal Database**.
3. Determine si la versión de su Centro de información de DB2 está anticuada comparando el nivel de la última imagen de documentación renovada con el

nivel de documentación que tenga instalado. El nivel de documentación que ha instalado aparece listado en la página de bienvenida del Centro de información de DB2.

4. Si se encuentra disponible una versión más reciente del Centro de información de DB2, descargue la última imagen renovada del *Centro de información de DB2* aplicable a su sistema operativo.
5. Para instalar la imagen renovada del *Centro de información de DB2*, siga las instrucciones proporcionadas en la página Web.

Conceptos relacionados:

- "Escenarios de instalación del Centro de información de DB2" en la página 814

Tareas relacionadas:

- "Invocación del Centro de información de DB2" en la página 821
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (UNIX)" en la página 816
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (Windows)" en la página 819

Visualización de temas en el idioma preferido en el Centro de información de DB2

El Centro de información de DB2 intenta visualizar los temas en el idioma especificado en las preferencias de navegador. Si un tema no se ha traducido al idioma preferido del usuario, el Centro de información de DB2 visualiza dicho tema en inglés.

Procedimiento:

Para visualizar temas en su idioma preferido en el navegador Internet Explorer:

1. En Internet Explorer, pulse el botón **Herramientas** —> **Opciones de Internet** —> **Idiomas...**. Se abrirá la ventana Preferencias de idioma.
2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.
   - Para añadir un nuevo idioma a la lista, pulse el botón **Agregar...**.

     Nota: La adición de un idioma no garantiza que el sistema tenga los fonts necesarios para visualizar los temas en el idioma preferido.
   - Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Subir** hasta que el idioma esté en primer lugar en la lista de idiomas.
3. Renueve la página a fin de visualizar el Centro de información de DB2 en su idioma preferido.

Para visualizar temas en su idioma preferido en el navegador Mozilla:

1. En Mozilla, seleccione el botón **Edit** —> **Preferences** —> **Languages**. Se visualizará el panel Languages en la ventana Preferences.
2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.
   - Para añadir un nuevo idioma a la lista, pulse el botón **Add...** a fin de seleccionar un idioma en la ventana Add Languages.

- Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Move Up** hasta que el idioma esté en primer lugar en la lista de idiomas.

3. Renueve la página a fin de visualizar el Centro de información de DB2 en su idioma preferido.

Conceptos relacionados:

- "Centro de información de DB2" en la página 812

Documentación PDF e impresa de DB2

Las tablas siguientes proporcionan los nombres oficiales de los manuales, los números de documento y los nombres de los archivos PDF. Para solicitar manuales en copia impresa, debe conocer el nombre oficial del manual. Para imprimir un archivo PDF, debe conocer el nombre del archivo PDF.

La documentación de DB2 está categorizada según las cabeceras siguientes:

- Información básica de DB2
- Información de administración
- Información para el desarrollo de aplicaciones
- Información de Business Intelligence
- Información de DB2 Connect
- Información de iniciación
- Información de aprendizaje
- Información sobre componentes opcionales
- Notas del release

Las tablas siguientes describen, para cada manual de la biblioteca de DB2, la información necesaria para solicitar la copia impresa o para imprimir o ver el PDF correspondiente al manual en cuestión. Se encuentra una descripción completa de cada uno de los manuales de la biblioteca de DB2 en el Centro de publicaciones de IBM de www.ibm.com/shop/publications/order

Información básica de DB2

La información de estos manuales es fundamental para todos los usuarios de DB2; encontrará útil esta información tanto si es programador o administrador de bases de datos como si trabaja con DB2 Connect, DB2 Warehouse Manager u otros productos de DB2.

Tabla 180. Información básica de DB2

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Universal Database Consulta de mandatos	SC10-3725	db2n0x81
IBM DB2 Universal Database Glosario	Sin número de documento	db2t0x81
IBM DB2 Universal Database Consulta de mensajes, Volumen 1	GC10-3728, no disponible en copia impresa	db2m1x81
IBM DB2 Universal Database Consulta de mensajes, Volumen 2	GC10-3729, no disponible en copia impresa	db2m2x81
IBM DB2 Universal Database Novedades	SC10-3734	db2q0x81

Información de administración

La información de estos manuales incluye los temas necesarios para diseñar, implementar y mantener de forma efectiva bases de datos de DB2, depósitos de datos y sistemas federados.

Tabla 181. Información de administración

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Universal Database Administration Guide: Planning	SC09-4822	db2d1x81
IBM DB2 Universal Database Administration Guide: Implementation	SC09-4820	db2d2x81
IBM DB2 Universal Database Administration Guide: Performance	SC09-4821	db2d3x81
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0x81
IBM DB2 Universal Database Data Movement Utilities Guide and Reference	SC09-4830	db2dmx81
IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference	SC09-4831	db2hax81
IBM DB2 Universal Database Data Warehouse Center Administration Guide	SC27-1123	db2ddx81
IBM DB2 Universal Database Consulta de SQL, Volumen 1	SC10-3730	db2s1x81
IBM DB2 Universal Database Consulta de SQL, Volumen 2	SC10-3731	db2s2x81
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0x81

Información para el desarrollo de aplicaciones

La información de estos manuales es de especial interés para los programadores de aplicaciones o programadores que trabajan con DB2 Universal Database (DB2 UDB). Hallará información acerca de los lenguajes y compiladores soportados, así como la documentación necesaria para acceder a DB2 UDB utilizando las diversas interfaces de programación soportadas, como, por ejemplo, SQL incorporado, ODBC, JDBC, SQLJ y CLI. Si utiliza el Centro de información de DB2, también podrá acceder a versiones HTML del código fuente para los programas de ejemplo.

Tabla 182. Información para el desarrollo de aplicaciones

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Universal Database Guía de desarrollo de aplicaciones: Creación y ejecución de aplicaciones	SC10-3733	db2axx81

Tabla 182. Información para el desarrollo de aplicaciones (continuación)

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Universal Database Guía de desarrollo de aplicaciones: Programación de aplicaciones de cliente	SC10-3723	db2a1x81
IBM DB2 Universal Database Guía de desarrollo de aplicaciones: Programación de aplicaciones de servidor	SC10-3724	db2a2x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2x81
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2adx81
IBM DB2 XML Extender Administración y programación	SC10-3750	db2sxx81

Información de Business Intelligence

La información de estos manuales describe cómo utilizar los componentes que mejoran las posibilidades de análisis y de depósito de datos de DB2 Universal Database.

Tabla 183. Información de Business Intelligence

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Warehouse Manager Standard Edition Information Catalog Center Administration Guide	SC27-1125	db2dix81
IBM DB2 Warehouse Manager Standard Edition Installation Guide	GC27-1122	db2idx81
IBM DB2 Warehouse Manager Standard Edition Managing ETI Solution Conversion Programs with DB2 Warehouse Manager	SC18-7727	iwhe1mstx80

Información de DB2 Connect

La información incluida en esta categoría describe cómo acceder a datos de servidores de sistema principal y de sistema medio utilizando DB2 Connect Enterprise Edition o DB2 Connect Personal Edition.

Tabla 184. Información de DB2 Connect

Nombre	Número de documento	Nombre de archivo PDF
IBM Connectivity Supplement	Sin número de documento	db2h1x81

Tabla 184. Información de DB2 Connect (continuación)

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Connect Guía rápida de iniciación para DB2 Enterprise Edition	GC10-3774	db2c6x81
IBM DB2 Connect Quick Beginnings for DB2 Connect Personal Edition	GC09-4834	db2c1x81
IBM DB2 Connect User's Guide	SC09-4835	db2c0x81

Información de iniciación

La información de esta categoría es útil cuando se van a instalar y configurar servidores, clientes y otros productos de DB2.

Tabla 185. Información de iniciación

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Universal Database Guía rápida de iniciación para clientes DB2	GC10-3775, no disponible en copia impresa	db2itx81
IBM DB2 Universal Database Guía rápida de iniciación para servidores DB2	GC10-3773	db2isx81
IBM DB2 Universal Database Guía rápida de iniciación para DB2 Personal Edition	GC10-3771	db2i1x81
IBM DB2 Universal Database Suplemento de instalación y configuración	GC10-3772, no disponible en copia impresa	db2iyx81
IBM DB2 Universal Database Guía rápida de iniciación para DB2 Data Links Manager	GC10-3726	db2z6x81

Información de aprendizaje

La información de aprendizaje presenta las características de DB2 y explica cómo realizar diversas tareas.

Tabla 186. Información de aprendizaje

Nombre	Número de documento	Nombre de archivo PDF
Guía de aprendizaje de Business Intelligence: Introducción al Centro de depósito de datos	Sin número de documento	db2tux81
Guía de aprendizaje de Business Intelligence: Lecciones ampliadas sobre depósito de datos	Sin número de documento	db2tax81
Information Catalog Center Tutorial	Sin número de documento	db2aix81
Guía de aprendizaje de Video Central para e-business	Sin número de documento	db2twx81
Guía de aprendizaje de Visual Explain	Sin número de documento	db2tvx81

Información sobre componentes opcionales

La información de esta categoría describe cómo trabajar con los componentes opcionales de DB2.

Tabla 187. Información sobre componentes opcionales

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Cube Views Guía y consulta	SC10–3868	db2aax81
IBM DB2 Query Patroller Guide: Installation, Administration and Usage Guide	GC09–7658	db2dwx81
IBM DB2 Spatial Extender and Geodetic Extender Guía del usuario y de consulta	SC10-3755	db2sbx81
IBM DB2 Universal Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0x82
DB2 Net Search Extender Administración y guía del usuario **Nota:** El HTML para este documento *no* se instala desde el CD de documentación HTML.	SH10-9305	N/D

Notas del release

Las notas del release proporcionan información adicional específica del release y nivel de FixPak del producto. Las notas del release también proporcionan resúmenes de las actualizaciones de la documentación que se han incorporado en cada release, actualización y FixPak.

Tabla 188. Notas del release

Nombre	Número de documento	Nombre de archivo PDF
Notas del release de DB2	Ver nota.	Ver nota.
Notas de instalación de DB2	Sólo disponible en el CD-ROM del producto.	No disponible.

Nota: Las Notas del release están disponibles en:

- XHTML y formato de texto, en los CD de los productos
- Formato PDF, en el CD de documentación PDF

Además, las partes de las Notas del release que tratan *Problemas conocidos y soluciones alternativas* e *Incompatibilidades entre releases* también aparecen en el Centro de información de DB2.

Para ver las Notas del release en formato de texto en las plataformas basadas en UNIX, consulte el archivo Release.Notes. Este archivo se encuentra en el directorio DB2DIR/Readme/%L, donde *%L* representa el nombre de entorno nacional y `DB2DIR` representa:

- En los sistemas operativos AIX: /usr/opt/db2_08_01
- En los otros sistemas operativos basados en UNIX: /opt/IBM/db2/V8.1

Conceptos relacionados:

- "Documentación y ayuda de DB2" en la página 811

Tareas relacionadas:

- "Impresión de manuales de DB2 desde archivos PDF" en la página 829
- "Solicitud de manuales de DB2 impresos" en la página 830
- "Invocación de ayuda según contexto desde una herramienta de DB2" en la página 830

Impresión de manuales de DB2 desde archivos PDF

Puede imprimir los manuales de DB2 desde los archivos PDF del *CD de documentación PDF de DB2*. Mediante la utilización de Adobe Acrobat Reader, puede imprimir el manual entero o un rango específico de páginas.

Prerrequisitos:

Asegúrese de que tiene instalado Adobe Acrobat Reader. Si ha de instalar Adobe Acrobat Reader, está disponible desde el sitio Web de Adobe en www.adobe.com

Procedimiento:

Para imprimir un manual de DB2 desde un archivo PDF:

1. Inserte el *CD de documentación PDF de DB2*. En sistemas operativos UNIX, monte el CD de documentación PDF de DB2. Consulte el manual *Iniciación rápida* para obtener detalles sobre cómo montar un CD en sistemas operativos UNIX.
2. Abra index.htm. El archivo de abre en una ventana de navegador.
3. Pulse el título del PDF que desee ver. El PDF se abrirá en Acrobat Reader.
4. Seleccione **Archivo** → **Imprimir** para imprimir cualquier parte que desee del manual.

Conceptos relacionados:

- "Centro de información de DB2" en la página 812

Tareas relacionadas:

- "Montaje del CD-ROM (AIX)" en la publicación *Guía rápida de iniciación para servidores DB2*
- "Cómo montar el CD-ROM (HP-UX)" en la publicación *Guía rápida de iniciación para servidores DB2*
- "Montaje del CD-ROM (Linux)" en la publicación *Guía rápida de iniciación para servidores DB2*
- "Solicitud de manuales de DB2 impresos" en la página 830
- "Montaje del CD-ROM (Entorno operativo Solaris)" en la publicación *Guía rápida de iniciación para servidores DB2*

Información relacionada:

- "Documentación PDF e impresa de DB2" en la página 824

Solicitud de manuales de DB2 impresos

Si prefiere utilizar manuales en copia impresa, puede solicitarlos de tres modos distintos.

Procedimiento:

Los manuales impresos se pueden solicitar en algunos países o regiones. Compruebe, en el sitio Web de publicaciones de IBM correspondiente a su país o región, si este servicio está disponible en su país o región. Cuando las publicaciones estén disponibles para su solicitud, puede realizar lo siguiente:

- Póngase en contacto con el distribuidor autorizado o representante de marketing de IBM. Para encontrar un representante local de IBM, consulte el directorio mundial de contactos de IBM en la página Web www.ibm.com/planetwide
- Llame al teléfono `1-800-879-2755`, si está en los EE.UU. o al `1-800-IBM-4YOU`, si está en Canadá.
- Visite el Centro de publicaciones de IBM en http://www.ibm.com/shop/publications/order. La capacidad de solicitar manuales desde el Centro de publicaciones de IBM puede no estar disponible en todos los países.

En el momento en que un producto de DB2 se encuentra disponible, los manuales impresos son los mismos que aparecen en formato PDF en el *CD de documentación PDF de DB2*. El contenido de los manuales impresos que se halla en el *CD del Centro de información de DB2* también es el mismo. No obstante, existe contenido adicional en el CD del Centro de información de DB2 que no aparece en ninguno de los manuales PDF (por ejemplo, rutinas de administración de SQL y ejemplos de HTML). No todos los manuales incluidos en el CD de documentación PDF de DB2 se pueden solicitar en copia impresa.

Nota: El Centro de información de DB2 se actualiza con mayor frecuencia que los manuales PDF o en copia impresa; instale las actualizaciones de la documentación a medida que estén disponibles o consulte el Centro de información de DB2 en http://publib.boulder.ibm.com/infocenter/db2help/ para obtener la información más actualizada.

Tareas relacionadas:

- "Impresión de manuales de DB2 desde archivos PDF" en la página 829

Información relacionada:

- "Documentación PDF e impresa de DB2" en la página 824

Invocación de ayuda según contexto desde una herramienta de DB2

La ayuda según contexto proporciona información sobre las tareas o controles que están asociados con una ventana, cuaderno, asistente o asesor determinado. La ayuda según contexto está disponible desde las herramientas de administración y desarrollo de DB2 que tienen interfaces gráficas de usuario. Existen dos tipos de ayuda según contexto:

- Ayuda a la que se accede mediante el botón **Ayuda** ubicado en cada ventana o cuaderno.

- Ventanas emergentes de información, que son ventanas que se visualizan cuando el cursor del ratón se coloca sobre un campo o control o cuando se selecciona un campo o control en una ventana, cuaderno, asistente o asesor y se pulsa F1.

El botón **Ayuda** proporciona acceso a la información de visión general, de prerrequisitos y de tareas. Las ventanas emergentes de información describen los campos y controles individuales.

Procedimiento:

Para invocar la ayuda según contexto:

- Para la ayuda de ventana y de cuaderno, inicie una de las herramientas de DB2 y, luego, abra cualquier ventana o cuaderno. Pulse el botón **Ayuda** situado en la esquina inferior derecha de la ventana o del cuaderno a fin de invocar la ayuda según contexto.

  También puede acceder a la ayuda según contexto desde el elemento de menú **Ayuda** situado en la parte superior de cada uno de los centros de herramientas de DB2.

  Para los asistentes y asesores, pulse en el enlace Visión general de tareas, de la primera página, si desea ver ayuda según contexto.
- Para obtener ayuda sobre controles individuales de una ventana o un cuaderno en una ventana emergente de información, pulse el control y, a continuación, pulse **F1**. La información emergente que contiene detalles sobre el control se visualizará en una ventana amarilla.

  Nota: Para visualizar ventanas emergentes de información simplemente manteniendo el cursor del ratón sobre un campo o control, seleccione el recuadro de selección **Visualizar automáticamente ventanas emergentes de información** en la página **Documentación** del cuaderno Valores de herramientas.

  Similar a las ventanas emergentes de información, la información emergente de diagnóstico es otra forma de ayuda según contexto; en ella se incluyen reglas para la entrada de datos. La información emergente de diagnóstico se visualiza en una ventana de color morado que aparece cuando se entran datos que no son válidos o que son insuficientes. La información emergente de diagnóstico puede aparecer para:
  - Campos obligatorios.
  - Campos cuyos datos tengan un formato preciso como, por ejemplo, un campo de fecha.

Tareas relacionadas:

- “Invocación del Centro de información de DB2” en la página 821
- “Invocación de la ayuda de mensajes desde el procesador de línea de mandatos” en la página 832
- “Invocación de la ayuda de mandatos desde el procesador de línea de mandatos” en la página 832
- “Invocación de la ayuda para estados de SQL desde el procesador de línea de mandatos” en la página 833
- “Acceso al Centro de información de DB2”
- “Cómo utilizar la ayuda de DB2 UDB”
- “Establecimiento de la ubicación para acceder al Centro de información de DB2”
- “Configuración del acceso a documentación y ayuda contextual de DB2”

Invocación de la ayuda de mensajes desde el procesador de línea de mandatos

La ayuda de mensajes describe la causa de un mensaje y describe la acción que se debe realizar en respuesta al error.

Procedimiento:

Para invocar la ayuda de mensajes, abra el procesador de línea de mandatos y entre:

```
? XXXnnnnn
```

donde *XXXnnnnn* representa un identificador de mensaje válido.

Por ejemplo, `? SQL30081` muestra la ayuda acerca del mensaje SQL30081.

Conceptos relacionados:

- "Introducción a los mensajes" en la publicación *Consulta de mensajes Volumen 1*

Información relacionada:

- "db2: Mandato de invocación del procesador de línea de mandatos" en la publicación *Consulta de mandatos*

Invocación de la ayuda de mandatos desde el procesador de línea de mandatos

La ayuda de mandatos explica la sintaxis de los mandatos del procesador de línea de mandatos.

Procedimiento:

Para invocar la ayuda de mandatos, abra el procesador de línea de mandatos y entre:

```
? mandato
```

donde *mandato* representa una palabra clave o el mandato completo.

Por ejemplo, `? catalog` visualiza ayuda para todos los mandatos CATALOG, mientras que `? catalog database` visualiza ayuda solamente para el mandato CATALOG DATABASE.

Tareas relacionadas:

- "Invocación de ayuda según contexto desde una herramienta de DB2" en la página 830
- "Invocación del Centro de información de DB2" en la página 821
- "Invocación de la ayuda de mensajes desde el procesador de línea de mandatos" en la página 832
- "Invocación de la ayuda para estados de SQL desde el procesador de línea de mandatos" en la página 833

Información relacionada:

- "db2: Mandato de invocación del procesador de línea de mandatos" en la publicación *Consulta de mandatos*

Invocación de la ayuda para estados de SQL desde el procesador de línea de mandatos

DB2 Universal Database devuelve un valor de SQLSTATE para las condiciones que pueden ser el resultado de una sentencia de SQL. La ayuda de SQLSTATE explica los significados de los estados de SQL y los códigos de las clases de estados de SQL.

Procedimiento:

Para invocar la ayuda para estados de SQL, abra el procesador de línea de mandatos y entre:

```
? sqlstate o ? código de clase
```

donde *sqlstate* representa un estado de SQL válido de cinco dígitos y *código de clase* representa los dos primeros dígitos del estado de SQL.

Por ejemplo, `? 08003` visualiza la ayuda para el estado de SQL 08003, y `? 08` visualiza la ayuda para el código de clase 08.

Tareas relacionadas:

- "Invocación del Centro de información de DB2" en la página 821
- "Invocación de la ayuda de mensajes desde el procesador de línea de mandatos" en la página 832
- "Invocación de la ayuda de mandatos desde el procesador de línea de mandatos" en la página 832

Guías de aprendizaje de DB2

Las guías de aprendizaje de DB2 ayudan a conocer los diversos aspectos de DB2 Universal Database. Las guías de aprendizaje proporcionan ejercicios con instrucciones paso a paso en las áreas de desarrollo de aplicaciones, ajuste del rendimiento de las consultas de SQL, trabajo con depósitos de datos, gestión de metadatos y desarrollo de servicios Web utilizando DB2.

Antes de empezar:

Puede ver las versiones XHTML de las guías de aprendizaje desde el Centro de información en http://publib.boulder.ibm.com/infocenter/db2help/.

Algunos ejercicios de las guías de aprendizaje utilizan datos o código de ejemplo. Consulte cada guía de aprendizaje para obtener una descripción de los prerrequisitos para las tareas específicas.

Guías de aprendizaje de DB2 Universal Database:

Pulse en el título de una guía de aprendizaje de la lista siguiente para ver esa guía de aprendizaje.

Guía de aprendizaje de Business Intelligence: Introducción al Centro de depósito de datos
Realizar tareas de introducción de depósito de datos utilizando el Centro de depósito de datos.

Guía de aprendizaje de Business Intelligence: Lecciones ampliadas sobre depósito de datos
Realizar tareas avanzadas de depósito de datos utilizando el Centro de depósito de datos.

Information Catalog Center Tutorial
Crear y gestionar un catálogo de información para localizar y usar metadatos utilizando el Centro de catálogos de información.

Guía de aprendizaje de Visual Explain
Analizar, optimizar y ajustar sentencias de SQL para obtener un mejor rendimiento al utilizar Visual Explain.

Información de resolución de problemas de DB2

Existe una gran variedad de información para la resolución de problemas y la determinación de problemas para ayudarle a utilizar los productos DB2®.

Documentación de DB2
La información de resolución de problemas se puede encontrar en todo el Centro de información de DB2, así como en todos los manuales PDF que componen la biblioteca de DB2. Puede consultar la rama sobre soporte y resolución de problemas, del árbol de navegación del Centro de información de DB2 (en el panel izquierdo de la ventana del navegador), para obtener un listado completo de la documentación de resolución de problemas de DB2.

Sitio Web de soporte técnico de DB2
Consulte el sitio Web de soporte técnico de DB2 si tiene problemas y desea obtener ayuda para encontrar las causas y las soluciones posibles. El sitio de soporte técnico tiene enlaces con las últimas publicaciones de DB2, notas técnicas, Informes autorizados de análisis del programa (APAR), FixPaks y el listado más reciente de códigos de error internos de DB2, además de otros recursos. Puede buscar en esta base de conocimiento para encontrar posibles soluciones a los problemas.

Para acceder al sitio Web de soporte de DB2, vaya a http://www.ibm.com/software/data/db2/udb/winos2unix/support

DB2 Problem Determination Tutorial Series (Serie de guías de aprendizaje para la determinación de problemas de DB2)
Consulte el sitio Web DB2 Problem Determination Tutorial Series para encontrar información sobre cómo identificar y resolver rápidamente los problemas que puedan surgir mientras trabaje con DB2. Una de las guías de aprendizaje ofrece una presentación de los recursos y las herramientas de determinación de problemas de DB2 disponibles y le ayuda a decidir cuándo utilizarlos. Otras de las guías de aprendizaje tratan temas relacionados como, por ejemplo, ″Determinación de problemas del motor de base de datos″, ″Determinación de problemas de rendimiento″ y ″Determinación de problemas de aplicaciones″.

Consulte el conjunto completo de guías de aprendizaje de determinación de problemas de DB2 en el sitio de soporte técnico de DB2 de http://www.ibm.com/software/data/support/pdm/db2tutorials.html

Conceptos relacionados:

- "Centro de información de DB2" en la página 812
- "Introduction to problem determination - DB2 Technical Support tutorial" en la publicación *Troubleshooting Guide*

Accesibilidad

Las características de accesibilidad ayudan a los usuarios con discapacidades físicas, por ejemplo movilidad o visión limitada, a utilizar los productos de software satisfactoriamente. La lista siguiente especifica las características de accesibilidad principales de los productos de DB2® Versión 8:

- Toda la funcionalidad de DB2 está disponible utilizando el teclado para la navegación en lugar del ratón. Si desea más información, consulte el apartado "Entrada de teclado y navegación".
- Puede personalizar el tamaño y color de los fonts en las interfaces de usuario de DB2. Si desea más información, consulte el apartado "Pantalla accesible".
- Los productos de DB2 dan soporte a aplicaciones de accesibilidad que utilizan la API de accesibilidad de Java™. Si desea más información, consulte el apartado "Compatibilidad con tecnologías de asistencia" en la página 836.
- La documentación de DB2 se proporciona en un formato accesible. Si desea más información, consulte el apartado "Documentación accesible" en la página 836.

Entrada de teclado y navegación

Entrada de teclado

Puede trabajar con las herramientas de DB2 utilizando solamente el teclado. Puede utilizar teclas o combinaciones de teclas para llevar a cabo operaciones que también se pueden realizar con el ratón. Las pulsaciones estándares del sistema operativo se utilizan para operaciones estándares del sistema operativo.

Para obtener más información sobre el uso de teclas o combinaciones de teclas al realizar operaciones, consulte Accesos directos y aceleradores del teclado.

Navegación de teclado

Puede navegar por la interfaz de usuario de las herramientas de DB2 mediante teclas o combinaciones de teclas.

Para obtener más información sobre el uso de teclas o combinaciones de teclas al navegar por las herramientas de DB2, consulte Accesos directos y aceleradores del teclado.

Foco del teclado

En los sistemas operativos UNIX®, se resalta el área de la ventana activa en la que las pulsaciones tendrán efecto.

Pantalla accesible

Las herramientas de DB2 presentan características que mejoran la accesibilidad de los usuarios con poca visión u otras discapacidades visuales. Estas mejoras de la accesibilidad incluyen soporte para propiedades de font personalizables.

Valores de font

Puede seleccionar el color, tamaño y font del texto en menús y ventanas de diálogo utilizando el cuaderno Valores de herramientas.

Para obtener más información sobre cómo especificar valores de font, consulte Modificación de fonts para menús y texto.

No dependencia del color

No es necesario distinguir los colores para utilizar cualquiera de las funciones de este producto.

Compatibilidad con tecnologías de asistencia

Las interfaces de las herramientas de DB2 dan soporte a la API de accesibilidad de Java, que le permite utilizar lectores de pantalla y otras tecnologías de asistencia con los productos de DB2.

Documentación accesible

La documentación de DB2 se proporciona en formato XHTML 1.0, que se puede visualizar en la mayoría de los navegadores Web. XHTML le permite visualizar la documentación de acuerdo con las preferencias de pantalla establecidas en el navegador. También permite utilizar lectores de pantalla y otras tecnologías de asistencia.

Los diagramas de sintaxis se proporcionan en formato decimal con puntos. Este formato sólo está disponible si se accede a la documentación en línea mediante un lector de pantalla.

Conceptos relacionados:

- "Diagramas de sintaxis en formato decimal con puntos" en la página 836

Tareas relacionadas:

- "Accesos directos y aceleradores del teclado"
- "Modificación de fonts para menús y texto"

Diagramas de sintaxis en formato decimal con puntos

Se proporcionan diagramas de sintaxis en formato decimal con puntos para los usuarios que acceden al Centro de información utilizando un lector de pantalla.

En formato decimal con puntos, cada elemento de sintaxis se escribe en una línea distinta. Si dos o más elementos de sintaxis siempre aparecen juntos (o siempre están ausentes los dos a la vez), pueden aparecer en la misma línea, puesto que se pueden considerar un elemento de sintaxis compuesto.

Cada línea empieza por un número decimal con puntos; por ejemplo, 3 ó 3.1 ó 3.1.1. Para oír estos números correctamente, asegúrese de que su lector de pantalla esté configurado para leer la puntuación. Todos los elementos de sintaxis que tienen el mismo número decimal con puntos (por ejemplo, todos los elementos de sintaxis que tienen el número 3.1) son alternativas mutuamente excluyentes. Si oye las líneas 3.1 USERID y 3.1 SYSTEMID, sabrá que la sintaxis puede incluir o USERID o SYSTEMID, pero no ambos.

El nivel de numeración decimal con puntos denota el nivel jerárquico. Por ejemplo, si un elemento de sintaxis con el número decimal con puntos 3 va seguido de una serie de elementos de sintaxis con el número decimal 3.1, todos los elementos de sintaxis con la numeración 3.1 son subordinados de los elementos de sintaxis identificados por el número 3.

Junto a los números decimales con puntos se utilizan determinados símbolos y palabras para añadir información sobre los elementos de sintaxis. A veces, estos símbolos y palabras pueden aparecer al principio del propio elemento. Para facilitar la identificación, si la palabra o el símbolo forman parte del elemento de sintaxis, van precedidos por una barra inclinada invertida (\). El símbolo * se puede utilizar junto a un número decimal con puntos para indicar que el elemento de sintaxis se repite. Por ejemplo, el elemento de sintaxis *FILE con el número decimal con puntos 3 adopta el formato 3 * FILE. El formato 3* FILE indica que el elemento de sintaxis FILE se repite. El formato 3* * FILE indica que el elemento de sintaxis * FILE se repite.

Los caracteres como las comas, que se utilizan para separar una serie de elementos de sintaxis, se muestran en la sintaxis justo antes de los elementos que separan. Estos caracteres pueden aparecer en la misma línea que cada elemento o en una línea distinta con el mismo número decimal con puntos que los elementos en cuestión. En la línea también puede aparecer otro símbolo que proporcione información sobre los elementos de sintaxis. Por ejemplo, las líneas 5.1*, 5.1 LASTRUN y 5.1 DELETE significan que si se utiliza más de uno de los elementos de sintaxis LASTRUN y DELETE, los elementos deben estar separados por comas. Si no hay ningún separador, suponga que utiliza un espacio en blanco para separar cada elemento de sintaxis.

Si un elemento de sintaxis va precedido del símbolo %, esto indica una referencia que está definida en cualquier otro lugar. La serie que aparece después del símbolo % es el nombre de un fragmento de sintaxis en lugar de un literal. Por ejemplo, la línea 2.1 %OP1 significa que se debe hacer referencia al fragmento de sintaxis separado OP1.

Junto a los números decimales con puntos se utilizan los símbolos y las palabras siguientes:

- ? indica un elemento de sintaxis opcional. Un número decimal con puntos seguido del símbolo ? indica que todos los elementos de sintaxis con un número decimal con puntos correspondiente y elementos de sintaxis subordinados son opcionales. Si sólo hay un elemento de sintaxis con un número decimal con puntos, el símbolo ? aparecerá en la misma línea que el elemento de sintaxis (por ejemplo, 5? NOTIFY). Si hay más de un elemento de sintaxis con un número decimal con puntos, el símbolo ? aparecerá en una línea propia, seguido de los elementos de sintaxis opcionales. Por ejemplo, si oye las líneas 5 ?, 5 NOTIFY y 5 UPDATE, sabrá que los elementos de sintaxis NOTIFY y UPDATE son opcionales; es decir, puede seleccionar uno o ninguno de dichos elementos. El símbolo ? es equivalente a una línea de desvío de un diagrama de vías.
- ! indica un elemento de sintaxis por omisión. Un número decimal con puntos seguido del símbolo ! y un elemento de sintaxis indica que el elemento de sintaxis es la opción por omisión para todos los elementos de sintaxis que comparten el mismo número decimal con puntos. Sólo uno de los elementos de sintaxis que comparten el mismo número decimal con puntos puede especificar un símbolo !. Por ejemplo, si oye las líneas 2? FILE, 2.1! (KEEP) y 2.1 (DELETE), sabrá que (KEEP) es la opción por omisión correspondiente a la palabra clave FILE. En este ejemplo, si incluye la palabra clave FILE pero no especifica ninguna opción, se aplicará la opción por omisión KEEP. También se aplicará una opción por omisión al siguiente número decimal con puntos más alto. En este ejemplo, si se omite la palabra clave FILE, se utiliza el valor por omisión FILE(KEEP). No obstante, si oye las líneas 2? FILE, 2.1, 2.1.1! (KEEP) y 2.1.1 (DELETE), la opción por omisión KEEP sólo se aplicará al siguiente número

decimal con puntos más alto, 2.1 (que no tiene una palabra clave asociada) y no se aplicará a 2? FILE. Si se omite la palabra clave FILE, no se utilizará nada.

- * indica un elemento de sintaxis que se puede repetir 0 o más veces. Un número decimal con puntos seguido del símbolo * indica que este elemento de sintaxis se puede utilizar cero o más veces; es decir, es opcional y se puede repetir. Por ejemplo, si oye la línea 5.1* data area, sabrá que puede incluir un área de datos, más de un área de datos o ningún área de datos. Si oye las líneas 3*, 3 HOST y 3 STATE, sabrá que puede incluir HOST, STATE, los dos juntos o ninguno de los dos.

  Notas:

  1. Si un número decimal con puntos tiene un asterisco (*) al lado y sólo hay un elemento con dicho número decimal con puntos, podrá repetir el mismo elemento más de una vez.
  2. Si un número decimal con puntos tiene un asterisco al lado y hay varios elementos que tienen dicho número decimal con puntos, podrá utilizar más de un elemento de la lista, pero no podrá utilizar los elementos más de una vez cada uno. En el ejemplo anterior, podría escribir HOST STATE pero no podría escribir HOST HOST.
  3. El símbolo * es equivalente a una línea de bucle de retorno de un diagrama de sintaxis de vías.

- + indica un elemento de sintaxis que se debe incluir una o más veces. Un número decimal con puntos seguido del símbolo + indica que este elemento de sintaxis se debe incluir una o más veces; es decir, se debe incluir como mínimo una vez y se puede repetir. Por ejemplo, si oye la línea 6.1+ data area, deberá incluir como mínimo un área de datos. Si oye las líneas 2+, 2 HOST y 2 STATE, sabrá que debe incluir HOST, STATE o ambos. De manera similar al símbolo *, el símbolo + sólo puede repetir un elemento determinado si éste es el único elemento que tiene el número decimal con puntos en cuestión. El símbolo +, al igual que el símbolo *, es equivalente a una línea de bucle de retorno de un diagrama de sintaxis de vías.

Conceptos relacionados:

- "Accesibilidad" en la página 835

Tareas relacionadas:

- "Accesos directos y aceleradores del teclado"

Información relacionada:

- "Lectura de los diagramas de sintaxis" en la página x

Certificación Common Criteria de productos DB2 Universal Database

Se está evaluando DB2 Universal Database para obtener la certificación Common Criteria en el nivel de garantía de evaluación 4 (EAL4). Para más información acerca de Common Criteria, consulte el sitio Web de Common Criteria en: http://niap.nist.gov/cc-scheme/.

Apéndice Q. Avisos

Es posible que IBM no comercialice en todos los países algunos productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en tramitación que afecten al tema tratado en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE.UU.

Para realizar consultas sobre licencias referentes a información de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país/región o escribir a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokio 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país/región en donde tales disposiciones sean incompatibles con la legislación local:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener inexactitudes técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán a las nuevas ediciones de la publicación. IBM puede efectuar, en cualquier momento y sin previo aviso, mejoras y cambios en los productos y programas descritos en esta publicación.

Las referencias hechas en esta publicación a sitios Web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen un aval de esos

sitios Web. La información contenida en esos sitios Web no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de dichos sitios Web.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información sobre él con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido éste) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADÁ

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos el pago de una tarifa.

El programa bajo licencia descrito en este documento y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Acuerdo de Cliente de IBM, el Acuerdo Internacional de Programas Bajo Licencia de IBM o cualquier acuerdo equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado en sistemas experimentales y no es seguro que estas mediciones sean las mismas en los sistemas disponibles comercialmente. Además, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios del presente manual deben verificar los datos aplicables para su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni ninguna otra afirmación referente a productos que no son de IBM. Las preguntas sobre las prestaciones de productos que no son de IBM deben dirigirse a los proveedores de esos productos.

Todas las declaraciones de intenciones de IBM están sujetas a cambio o cancelación sin previo aviso, y sólo representan objetivos.

Este manual puede contener ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente fortuita.

LICENCIA DE COPYRIGHT:

Este manual puede contener programas de aplicaciones de ejemplo escritos en lenguaje fuente, que muestran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo como desee, sin pago alguno a IBM, con la intención de desarrollar, utilizar, comercializar o distribuir programas de aplicaciones de acuerdo con la interfaz de programación de aplicaciones correspondiente a la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por lo tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas.

Cada copia o parte de estos programas de ejemplo o cualquier trabajo derivado debe incluir una nota de copyright como la siguiente:

© (*nombre de la empresa*) (*año*). Partes de este código proceden de programas de ejemplo de IBM Corp. © Copyright IBM Corp. *_entre el o los años_*. Reservados todos los derechos.

Marcas registradas

Los términos siguientes son marcas registradas de International Business Machines Corporation en los EE.UU. y/o en otros países y se han utilizado como mínimo en uno de los documentos de la biblioteca de documentación de DB2 UDB.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeriesLAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

Los términos siguientes son marcas registradas de otras empresas y se han utilizado como mínimo en uno de los documentos de la biblioteca de documentación de DB2 UDB:

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los EE.UU. y/o en otros países.

Intel y Pentium son marcas registradas de Intel Corporation en los EE.UU. y/o en otros países.

Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en los EE.UU. y/o en otros países.

UNIX es marca registrada de The Open Group en los EE.UU. y/o en otros países.

Otros nombres de empresas, productos o servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

Caracteres Especiales

A

B

C

CH

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Cómo ponerse en contacto con IBM

En los EE.UU., puede ponerse en contacto con IBM llamando a uno de los siguientes números:

- 1-800-IBM-SERV (1-800-426-7378) para servicio al cliente
- 1-888-426-4343 para obtener información sobre las opciones de servicio técnico disponibles
- 1-800-IBM-4YOU (426-4968) para marketing y ventas de DB2

En Canadá, puede ponerse en contacto con IBM llamando a uno de los siguientes números:

- 1-800-IBM-SERV (1-800-426-7378) para servicio al cliente
- 1-800-465-9600 para obtener información sobre las opciones de servicio técnico disponibles
- 1-800-IBM-4YOU (1-800-426-4968) para marketing y ventas de DB2

Para localizar una oficina de IBM en su país o región, consulte IBM Directory of Worldwide Contacts en el sitio Web http://www.ibm.com/planetwide

Información sobre productos

La información relacionada con productos DB2 Universal Database se encuentra disponible por teléfono o a través de la World Wide Web en el sitio http://www.ibm.com/software/data/db2/udb

Este sitio contiene la información más reciente sobre la biblioteca técnica, pedidos de manuales, descargas de productos, grupos de noticias, FixPaks, novedades y enlaces con recursos de la Web.

Si vive en los EE.UU., puede llamar a uno de los números siguientes:

- 1-800-IBM-CALL (1-800-426-2255) para solicitar productos u obtener información general.
- 1-800-879-2755 para solicitar publicaciones.

Para obtener información sobre cómo ponerse en contacto con IBM desde fuera de los EE.UU., vaya a la página IBM Worldwide en el sitio www.ibm.com/planetwide

SC10-3730-01

Spine information:

IBM® DB2 Universal Database™ Consulta de SQL, Volumen 1 Versión 8.2