IBM® DB2® Spatial Extender

Guía y consulta del usuario

Versión 7

SC10-3528-00

IBM® DB2® Spatial Extender

Guía y consulta del usuario

Versión 7

SC10-3528-00

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general incluida en el tema “Avisos” en la página 329. .

Este manual es la traducción del original inglés *IBM DB2 Spatial Extender User's Guide and Reference Version 7, SC27-0701-00*.

Contenido

Figuras

Tablas

Acerca de este manual

Esta manual está dividido en dos partes. La primera parte contiene información conceptual sobre DB2 Spatial Extender y explica cómo instalar, configurar, administrar y programar para DB2 Spatial Extender en sistemas Windows NT y AIX. La segunda parte consta de información de consulta sobre procedimientos almacenados, geometrías, funciones, mensajes y vistas de catálogos que utiliza con DB2 Spatial Extender.

A quién va dirigido este manual

Este manual está destinado a administradores que configuran el entorno espacial y a programadores de aplicaciones que desarrollan aplicaciones con datos espaciales.

Convenios

En este manual se utilizan los siguientes convenios de resaltado:

Letra negrita
Indica mandatos y controles de la interfaz gráfica de usuario (GUI) (por ejemplo, nombres de campos, nombres de carpetas, opciones de menús).

`Letra monoespaciado`
Indica ejemplos de codificación o de texto que escribe el usuario.

Letra cursiva
Indica variables que el usuario debe sustituir por un valor. El tipo cursiva indica también títulos de manuales y enfatiza palabras.

TIPO MAYÚSCULAS
Indica palabras clave de SQL y nombres de objetos (por ejemplo, tablas, vistas y servidores).

Cómo enviar sus comentarios

La información que envía ayuda a IBM a ofrecer información de calidad. Por favor, envíe los comentarios que desee sobre este manual o sobre otra documentación de DB2. Puede utilizar cualquiera de los siguientes métodos para transmitir sus comentarios:

- Envíe sus comentarios desde la Web. Puede acceder al formulario de comentarios del lector en línea de IBM Data Management en la dirección http://www.ibm.com/software/data/rcf

- Envíe sus comentarios por correo electrónico a comments@vnet.ibm.com. Asegúrese de incluir el nombre del producto, el número de versión del producto y el nombre y el número de pieza del manual (si se aplica). Si efectúa comentarios sobre un texto específico, por favor, indique dónde se encuentra el texto (por ejemplo, un capítulo y un título de sección, un número de tabla, un número de página o un título de un tema de ayuda).

Parte 1. Utilización de DB2 Spatial Extender

Capítulo 1. Acerca de DB2 Spatial Extender

Este capítulo contiene una introducción a DB2 Spatial Extender y explica su finalidad y los datos que procesa e indica cómo utilizarlo. Este capítulo concluye con una guía rápida sobre el resto del manual.

La finalidad de DB2 Spatial Extender

DB2 Spatial Extender sirve para crear un *sistema de información geográfica* (GIS): un grupo de objetos, datos y aplicaciones que le ayudan a generar y a analizar información espacial sobre funciones geográficas. Las *funciones geográficas* incluyen los objetos que comprenden la superficie de la tierra y los objetos que la ocupan. Forman tanto el entorno natural (como por ejemplo ríos, bosques, montañas y desiertos) como el entorno cultural (ciudades, residencias, edificios de oficinas, monumentos, etc.).

La *información espacial* incluye hechos como:

- La ubicación de funciones geográficas con respecto a sus alrededores (por ejemplo, puntos dentro de la ciudad donde se encuentran hospitales y clínicas o la proximidad de las zonas residenciales de una ciudad con respecto a zonas de terremotos locales)
- Modos en que las funciones geográficas se relacionan entre sí (por ejemplo, información sobre que un determinado sistema de ríos queda enclavado en una determinada región, o sobre que ciertos puentes de dicha región cruzan los afluentes del sistema de ríos)
- Las medidas que se aplican a una o más funciones geográficas (por ejemplo, la distancia entre un edificio de oficinas y el límite del terreno o la longitud de un perímetro a vista de pájaro)

La información espacial, por sí misma o en combinación con la salida de un sistema tradicional de gestión de bases de datos relacionales (RDBMS), le puede ayudar a diseñar proyectos y a tomar decisiones empresariales o políticas. Por ejemplo, supongamos que el responsable de prestaciones sociales de un distrito tiene que comprobar cuáles de los candidatos y receptores de las prestaciones sociales viven realmente dentro del área a la que se aplican las prestaciones. DB2 Spatial Extender puede deducir esta información a partir de la ubicación del área y de las direcciones de los candidatos y de los receptores.

Supongamos ahora que el propietario de una cadena de restaurantes desea comenzar el negocio en ciudades cercanas. Para determinar dónde abrir nuevos restaurantes, el propietario necesita respuestas a preguntas como: ¿En

qué puntos de la ciudad se concentra la clientela que suele frecuentar mis restaurantes? ¿Dónde están las principales carreteras? ¿En qué puntos es más baja la tasa de criminalidad? ¿Dónde se encuentran los restaurantes de la competencia? DB2 Spatial Extender puede generar información espacial en representaciones visuales para responder a estas preguntas y el RDBMS subyacente puede generar etiquetas y texto que expliquen las representaciones visuales.

En este manual aparecen muchos otros ejemplos de aplicación de DB2 Spatial Extender, especialmente en el “Capítulo 7. Recuperación y análisis de información espacial” en la página 55, en el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59 y en el “Capítulo 14. Funciones espaciales para consultas SQL” en la página 169.

Datos que representan funciones geográficas

Esta sección contiene una visión general de los datos que puede generar, almacenar y manipular para obtener información espacial. Los temas de esta sección son:

- Cómo los datos representan funciones geográficas
- La naturaleza de los datos espaciales
- Modos de generar datos espaciales

Cómo los datos representan funciones geográficas

En DB2 Spatial Extender, una función geográfica se puede representar mediante una fila de una tabla o vista o mediante una parte de dicha fila. Por ejemplo, consideremos dos de las funciones geográficas que se mencionan en el tema “La finalidad de DB2 Spatial Extender” en la página 3, edificios de oficinas y residencias. En la Figura 1 en la página 5, cada fila de la tabla BRANCHES representa una sucursal de un banco. Como variación, cada fila de la tabla CUSTOMERS de la Figura 1 en la página 5, considerada en conjunto, representa un cliente del banco. Sin embargo, se puede considerar que parte de cada fila—en concreto, las celdas que contienen la dirección de un cliente—representa la residencia del cliente.

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141	A	A

Figura 1. Fila de tabla que representa una función geográfica; fila de tabla cuyos datos de dirección representan una función geográfica. La fila de datos de la tabla BRANCHES representa una sucursal de un banco. Las celdas de datos de dirección de la tabla CUSTOMERS representan la residencia de un cliente. Los nombres y direcciones de ambas tablas son ficticios.

Las tablas de la Figura 1 contienen datos que identifican y describen las sucursales del banco y sus clientes. Dichos datos se denominan *datos de atributo.*

Un subconjunto de los datos de atributo—los valores que representan direcciones de sucursales y de clientes—se pueden convertir en valores que aportan información espacial. Por ejemplo, tal como se muestra en la Figura 1, la dirección de una sucursal es 92467 Airzone Blvd., San Jose CA 95141. La dirección de un cliente es 9 Concourt Circle, San Jose CA 95141. DB2 Spatial Extender puede convertir estas direcciones en valores que indiquen dónde están situadas la sucursal y la residencia del cliente con respecto a sus alrededores. La Figura 2 muestra las tablas BRANCHES y CUSTOMERS con nuevas columnas designadas para contener dichos valores.

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141		A	A

Figura 2. Tablas con columnas espaciales añadidas. En cada tabla, la columna LOCATION contendrá coordenadas correspondientes a las direcciones.

Cuando se utilizan direcciones e identificadores parecidos como punto inicial para la información espacial, reciben el nombre de *datos fuente.* Puesto que los valores obtenidos a partir de los mismos aportan información espacial, estos

valores obtenidos se denominan *datos espaciales*. La siguiente sección describe los datos espaciales y contiene una introducción a sus tipos de datos asociados.

La naturaleza de los datos espaciales

Muchos datos espaciales están formados por coordenadas. Una *coordenada* es un número que indica una posición relativa a un punto de referencia. Por ejemplo, las latitudes son coordenadas que indican posiciones con respecto al ecuador. Las longitudes son coordenadas que indican posiciones con respecto al meridiano de Greenwich. De este modo, la posición del Parque Nacional de Yellowstone se puede definir por su latitud (44,45 grados al norte del ecuador) y su longitud (110,40 grados al oeste del meridiano de Greenwich).

Las latitudes, las longitudes, sus puntos de referencia y otros parámetros asociados reciben, en conjunto, el nombre de *sistema de coordenadas*. También hay sistemas de coordenadas basados en valores que no son latitud y longitud. Estos sistemas de coordenadas tienen sus propias medidas de posición, puntos de referencia y parámetros distintivos adicionales.

El elemento más sencillo de datos espaciales consiste en dos coordenadas que definen la posición de una sola función geográfica. (Un *elemento de datos* es el valor o valores que ocupan una celda de una tabla relacional.) Un elemento de datos espaciales más extenso consta de varias coordenadas que definen un recorrido lineal como el que puede formar un camino o un río. Un tercer tipo consiste en coordenadas que definen el perímetro de un área; por ejemplo, los límites de una parcela de tierra o de la zona que queda inundada tras la crecida de un río. Estos y otros tipos de elementos de datos espaciales a los que da soporte DB2 Spatial Extender se describen con detalle en el “Capítulo 13. Geometrías y funciones espaciales asociadas” en la página 131.

Cada elemento de datos espaciales es una instancia de un tipo de datos espaciales. El tipo de datos correspondiente a dos coordenadas que marcan una ubicación es ST_Point; el tipo de datos correspondiente a coordenadas que definen recorridos lineales es ST_LineString; y el tipo de datos correspondiente a coordenadas que definen perímetros es ST_Polygon. Estos tipos, junto con otros tipos de datos correspondientes a datos espaciales, son tipos estructurados que pertenecen a una sola jerarquía. Para ver una visión general de la jerarquía, consulte el tema “Acerca de los tipos de datos espaciales” en la página 33.

De dónde proceden los datos espaciales

Puede obtener datos espaciales:

- A partir de datos de atributo
- A partir de otros datos espaciales
- Importándolos

Utilización de datos de atributo como datos fuente

DB2 Spatial Extender puede obtener datos espaciales a partir de datos de atributo, como direcciones (tal como se menciona en el tema "Cómo los datos representan funciones geográficas" en la página 4). Este proceso se denomina *geocodificación*. Para ver la secuencia implicada, considere que la Figura 2 en la página 5 es una imagen de "antes" y que la Figura 3 es una imagen de "después". La Figura 2 en la página 5 muestra que la tabla BRANCHES y la tabla CUSTOMERS tienen una columna vacía designada para datos espaciales. Supongamos que DB2 Spatial Extender geocodifica las direcciones de estas tablas para obtener coordenadas correspondientes a las direcciones y coloca las coordenadas en las columnas. La Figura 3 ilustra este resultado.

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	1653 3094

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141	953 1527	A	A

Figura 3. Tablas que incluyen datos espaciales obtenidos a partir de datos fuente. La columna LOCATION de la tabla CUSTOMERS contiene coordenadas que un geocodificador ha obtenido a partir de la dirección de las columnas ADDRESS, CITY, STATE y ZIP. De forma similar, la columna LOCATION de la tabla BRANCHES contiene coordenadas que el geocodificador ha obtenido a partir de la dirección de las columnas ADDRESS, CITY, STATE y ZIP de esta tabla. Este ejemplo es ficticio; se muestran coordenadas simuladas, no reales.

DB2 Spatial Extender utiliza una función, denominada un *geocodificador*, para convertir datos de atributo en datos espaciales y para colocar estos datos espaciales en columnas de tablas. Para obtener más información sobre geocodificadores, consulte el tema "Acerca de la geocodificación" en la página 41.

Utilización de otros datos espaciales como datos fuente

Se pueden generar datos espaciales no sólo a partir de datos de atributo, sino también a partir de otros datos espaciales. Por ejemplo, supongamos que el banco cuyas sucursales están definidas en la tabla BRANCHES desea saber el número de clientes situados a menos de cinco millas de cada sucursal. Para que el banco pueda obtener esta información de la base de datos, debe suministrar a la misma la definición de la zona que queda dentro de un radio de cinco millas alrededor de cada sucursal. Una función de DB2 Spatial Extender, ST_Buffer, puede crear esta definición. Utilizando las coordenadas de cada sucursal como entrada, ST_Buffer puede generar las coordenadas que

marcan los perímetros de las zonas deseadas. La Figura 4 muestra la tabla BRANCHES con información suministrada por ST_Buffer.

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	SALES_AREA
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	1653 3094	1002 2001, 1192 3564, 2502 3415, 1915 3394, 1002 2001

Figura 4. Tabla que incluye nuevos datos espaciales obtenidos a partir de datos espaciales existentes. La función ST_Buffer ha obtenido las coordenadas de la columna SALES_AREA a partir de las coordenadas de la columna LOCATION. Al igual que las coordenadas de la columna LOCATION, las de la columna SALES_AREA son también simuladas; no son reales.

Además de ST_Buffer, DB2 Spatial Extender ofrece otras funciones para obtener nuevos datos espaciales a partir de datos espaciales existentes. Para ver descripciones de ST_Buffer y de estas otras funciones, consulte el tema "Funciones que generan geometrías nuevas a partir de geometrías existentes" en la página 158.

Importación de datos espaciales

Un tercer modo de obtener datos espaciales consiste en importarlos de archivos que estén en uno de los formatos a los que da soporte DB2 Spatial Extender. Para ver descripciones de estos formatos, consulte el "Capítulo 16. Formatos de archivos para datos espaciales" en la página 303. Estos archivos contienen datos que se suelen aplicar a correlaciones: seguimientos del censo, zonas que quedan inundadas tras la crecida de un río, fallas, etc. Al utilizar dichos datos en combinación con datos espaciales generados por el usuario, puede aumentar la información geográfica disponible. Por ejemplo, si un departamento de obras públicas tuviera que determinar a qué peligros se expone una comunidad residencial, podría utilizar ST_Buffer para definir una zona alrededor de la comunidad. El departamento de obras públicas podría importar datos sobre zonas que quedan inundadas tras la crecida de un río o fallas para ver cuáles de estas áreas de problemas se aplican a la zona.

Cómo crear y utilizar un GIS de DB2 Spatial Extender

Para crear un GIS de DB2 Spatial Extender debe definir DB2 Spatial Extender y desarrollar proyectos GIS dentro de los entornos combinados de DB2 Spatial Extender y su DB2 RDBMS subyacente. Puede utilizar el GIS implantando estos proyectos; es decir, generando y analizando la información para la que están diseñados suministrar—tanto espacial como tradicional—. Para ello debe llevar a cabo varios grupos de tareas. Esta sección contiene una introducción a las interfaces con las que puede realizar estas tareas, ofrece una visión general de las tareas y presenta un escenario que las ilustra.

Interfaces ante DB2 Spatial Extender y funciones asociadas

Esta sección describe las interfaces mediante las que puede crear un GIS de DB2 Spatial Extender (es decir, definir recursos para el mismo, obtener datos espaciales, etc.) y utilizarlo (es decir, generar y analizar información sobre funciones geográficas).

Puede crear un GIS de DB2 Spatial Extender:

- Utilizando las opciones de menús y ventanas de DB2 Spatial Extender del Centro de datos de DB2. Para obtener instrucciones, consulte:
  - "Capítulo 3. Configuración de recursos" en la página 23
  - "Capítulo 4. Cómo definir columnas espaciales, registrarlas como capas y habilitar un geocodificador para que las mantenga" en la página 33
  - "Capítulo 5. Cómo llenar columnas espaciales" en la página 41
  - "Capítulo 6. Creación de índices espaciales" en la página 53
- Ejecutando un programa de aplicación que llame a procedimientos almacenados de DB2 Spatial Extender. Para obtener directrices sobre cómo desarrollar un programa de este tipo, consulte el "Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 59.
- Utilizando el Centro de control y un programa de aplicación. Por ejemplo, puede utilizar el Centro de control para invocar el geocodificador por omisión. Si, además, desea utilizar otro geocodificador, primero debe registrarlo ante DB2 Spatial Extender invocando el procedimiento almacenado db2gse.gse_register_gc en un programa de aplicación. (Para obtener información sobre geocodificadores que no son el geocodificador por omisión, consulte el tema "Acerca de la geocodificación" en la página 41. Para obtener información sobre el procedimiento almacenado db2gse.gse_register_gc, consulte el tema "db2gse.gse_register_gc" en la página 93.)
- Utilizando el Centro de control, un programa de aplicación, o ambos junto con otras interfaces. Por ejemplo, para crear una tabla para albergar datos generados por una función espacial, como un geocodificador, podría utilizar el procesador de línea de mandatos o las interfaces del Centro de control.

Puede utilizar un GIS de DB2 Spatial Extender:

- Representando información gráficamente con un geoexaminador; por ejemplo ArcExplorer, que ofrece el Environmental Systems Research Institute (ESRI)
- Sometiendo consultas de SQL de forma explícita desde el Centro de control de DB2 o el procesador de línea de mandatos
- Sometiendo consultas de SQL desde un programa de aplicación

Tareas que debe realizar para crear y utilizar un GIS de DB2 Spatial Extender

Esta sección contiene una visión general de las tareas a realizar para crear y utilizar un GIS de DB2 Spatial Extender. Las tareas a realizar para crear el GIS incluyen la configuración de DB2 Spatial Extender y el desarrollo de proyectos GIS. Las tareas a realizar para utilizar el GIS incluyen la implantación de los proyectos. Esta visión general empieza por la configuración de DB2 Spatial Extender y luego pasa por el desarrollo e implantación de un proyecto GIS. Esta sección concluye indicando cómo las tareas descritas en la visión general pueden variar en la realidad.

Configuración de DB2 Spatial Extender

Para configurar DB2 Spatial Extender:

1. Planifique y prepare (decida qué proyectos GIS desea desarrollar y qué base de datos debe habilitar para DB2 Spatial Extender, seleccione personal para administrar DB2 Spatial Extender y desarrollar los proyectos, etc).
2. Instale DB2 Spatial Extender.
3. Coloque recursos para dar soporte a los proyectos GIS; por ejemplo:

   Recursos suministrados por DB2 Spatial Extender
   Estos incluyen un catálogo del sistema, tipos de datos espaciales, funciones espaciales (incluido un geocodificador por omisión), etc. La tarea de configurar estos recursos se denomina *habilitación de la base de datos para operaciones espaciales.*

   Geocodificadores desarrollados por usuarios, proveedores o ambos
   El geocodificador por omisión convierte direcciones de Estados Unidos en datos espaciales. Su organización y otras pueden suministrar geocodificadores que conviertan direcciones de otros países y otros tipos de datos de atributo en datos espaciales.

Para obtener instrucciones sobre cómo instalar DB2 Spatial Extender, consulte el “Capítulo 2. Instalación de DB2 Spatial Extender” en la página 17. Para obtener instrucciones sobre cómo utilizar el Centro de control para colocar recursos, consulte el “Capítulo 3. Configuración de recursos” en la página 23. Para ver directrices sobre cómo utilizar un programa de aplicación para esta finalidad, consulte el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59. Para ver un escenario que ilustre la tarea de configurar DB2 Spatial Extender, consulte el tema “Un sistema para integrar datos espaciales y tradicionales” en la página 13.

Desarrollo e implantación de un proyecto GIS

Para desarrollar e implantar un proyecto GIS:

1. Planifique y prepare (defina objetivos del proyecto, decida qué tablas y datos necesita, determine qué sistema o sistemas de coordenadas utilizar, etc.)

2. Decida qué sistema o sistemas de referencia espacial utilizar. Los valores de las coordenadas suelen incluir enteros positivos, números negativos y números decimales. Sin embargo, DB2 Spatial Extender debe almacenar todos los valores de coordenadas en formato de enteros positivos. Un *sistema de referencias espaciales* es un grupo de parámetros que define el modo en que los números negativos y decimales se deben convertir en enteros positivos para que DB2 Spatial Extender los pueda almacenar. Después de decidir qué sistema de coordenadas utilizar para una columna espacial, tiene que especificar el sistema de referencias espaciales según el que se llevará a cabo la conversión necesaria correspondiente a dicha columna. Si un sistema de referencias espaciales se ajusta a sus requisitos, puede utilizarlo; si no es así, puede crear uno.
3. Defina una o más columnas para que contengan datos espaciales, regístrelas ante DB2 Spatial Extender y habilite un geocodificador para mantenerlas de forma automática.

   Para registrar una columna espacial hay que registrarla en el catálogo de DB2 Spatial Extender. Desde el momento en que la registra se denomina *capa*, porque la información generada a partir de la misma añadirá un estrato, o capa, al paisaje geográfico virtual que genera GIS. Una vez registrada, puede realizar operaciones espaciales sobre la misma; por ejemplo, puede llenarla y definir en la misma un índice espacial.
4. Llene columnas espaciales;
   - Para un proyecto que necesite un geocodificador, defina parámetros para el mismo. Luego ejecútelo de modo que, en una sola operación, geocodifique todos los datos fuente disponibles y cargue las coordenadas resultantes en una capa.
   - Para un proyecto que necesite que se importen datos espaciales, importe los datos.
5. Facilite acceso a columnas espaciales. En concreto, esto implica definir índices que permitan a DB2 acceder a datos espaciales de forma rápida y definir vistas que permitan a los usuarios recuperar datos interrelacionados de forma eficiente. Después de definir una vista de este tipo, tiene que registrar sus columnas espaciales como capas.
6. Genere y analice información espacial e información comercial relacionada. Esto implica consultar columnas espaciales y columnas de atributo relacionadas. En dichas consultas, incluirá funciones de DB2 Spatial Extender que devuelven una gran variedad de información; por ejemplo, la distancia mínima entre dos funciones geográficas o coordenadas que definan un área que quede alrededor de una función geográfica. Para obtener información sobre la función que devuelve dichas coordenadas, ST_Buffer, consulte el tema “Utilización de otros datos espaciales como datos fuente” en la página 7 y el tema “ST_Buffer” en la página 199. Para ver ejemplos de consultas que utilizan funciones espaciales, consulte el

"Capítulo 7. Recuperación y análisis de información espacial" en la página 55 y el "Capítulo 14. Funciones espaciales para consultas SQL" en la página 169.

Para ver instrucciones sobre cómo utilizar el Centro de control para llevar a cabo tareas relacionadas con el desarrollo de un proyecto GIS, consulte:

- "Capítulo 3. Configuración de recursos" en la página 23
- "Capítulo 4. Cómo definir columnas espaciales, registrarlas como capas y habilitar un geocodificador para que las mantenga" en la página 33
- "Capítulo 5. Cómo llenar columnas espaciales" en la página 41
- "Capítulo 6. Creación de índices espaciales" en la página 53

Para ver instrucciones sobre cómo utilizar el Centro de control para implantar un proyecto GIS, consulte el "Capítulo 7. Recuperación y análisis de información espacial" en la página 55.

Para ver instrucciones sobre cómo utilizar un programa de aplicación para desarrollar e implantar un proyecto GIS, consulte el "Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 59.

Para ver un escenario que ilustre esta tarea, consulte "Un proyecto para establecer sucursales y ajustar primas" en la página 13.

Cómo pueden variar los grupos de tareas

El grupo de tareas que puede llevar a cabo para crear y utilizar un GIS de DB2 Spatial Extender puede variar en contenido y secuencia, en función de sus requisitos y de las interfaces que utilice. Por ejemplo, considere las tareas de definir columnas que contienen datos espaciales, registrarlas como capas y habilitar un geocodificador para que las mantenga de forma automática. Con el Centro de control, puede llevar a cabo estas tareas en conjunto, desde una sola ventana. Sin embargo, si invoca procedimientos almacenados desde un programa, puede realizar estas tareas por separado y las puede planificar como desee.

Escenario: una compañía de seguros actualiza su GIS

Esta sección presenta un escenario para ilustrar los grupos de tareas descritos en la sección anterior.

El entorno de sistemas de información de la compañía de seguros Safe Harbor Real Estate incluye un sistema DB2 Universal Database y un sistema de gestión de bases de datos GIS separado. Hasta cierto punto, las consultas pueden recuperar combinaciones de datos procedentes de los dos sistemas. Por ejemplo, una tabla de DB2 almacena información sobre ingresos y una tabla de GIS almacena las ubicaciones de las sucursales de la compañía. Por lo tanto, es posible encontrar las ubicaciones de las sucursales que aportan ingresos de determinadas cantidades. Pero los datos procedentes de los dos

sistemas no se pueden integrar (por ejemplo, los usuarios no pueden unir columnas de DB2 con columnas de GIS) y los servicios de DB2, como la optimización de consultas, no están disponibles para GIS. Para solucionar estos problemas, Safe Harbor adquiere DB2 Spatial Extender y establece un nuevo departamento de desarrollo de GIS. Las siguientes secciones describen cómo el departamento configura DB2 Spatial Extender y lleva a cabo su primer proyecto.

Un sistema para integrar datos espaciales y tradicionales

Para configurar DB2 Spatial Extender, el departamento de desarrollo de GIS de Safe Harbor actúa del siguiente modo:

1. El departamento se prepara para incluir DB2 Spatial Extender en su entorno DB2. Por ejemplo:
   a. El equipo de dirección del departamento asigna a un equipo de administración espacial la tarea de instalar e implantar DB2 Spatial Extender y a un equipo de análisis espacial la tarea de generar y analizar información espacial.
   b. Puesto que las decisiones comerciales de Safe Harbor se basan principalmente en los requisitos de los clientes, el equipo de dirección decide instalar DB2 Spatial Extender en la base de datos que contiene información sobre los clientes. La mayor parte de esta información se almacena en una tabla denominada CUSTOMERS.

      Para hacer referencia fácilmente a la base de datos seleccionada, los miembros del departamento de desarrollo de GIS la llaman *base de datos GIS*. Sin embargo, saben que no está reservada únicamente para proyectos GIS; las aplicaciones no espaciales la pueden seguir utilizando, como antes.
2. El equipo de administración espacial instala DB2 Spatial Extender.
3. El equipo de administración espacial configura recursos que necesitarán los proyectos GIS:
   - El equipo utiliza el Centro de control para suministrar los recursos que habilitan la base de datos GIS para operaciones espaciales. Estos recursos incluyen el catálogo de DB2 Spatial Extender, tipos de datos espaciales, funciones espaciales, etc.
   - Puesto que Safe Harbor está empezando a realizar operaciones comerciales en Canadá, el equipo de administración espacial empieza solicitando a los proveedores canadienses geocodificadores para convertir las direcciones de Canadá en datos espaciales.

Un proyecto para establecer sucursales y ajustar primas

Para llevar a cabo su primer proyecto GIS bajo DB2 Spatial Extender, el equipo de desarrollo de GIS actúa del siguiente modo:

1. El departamento se prepara para desarrollar el proyecto; por ejemplo:
   - El equipo de dirección define objetivos para el proyecto:

- Para determinar dónde establecer nuevas sucursales.
- Para ajustar primas en función de la proximidad de los clientes a áreas peligrosas (áreas con alto nivel de accidentes de tráfico, áreas con alto nivel de criminalidad, zonas que se inundan tras la crecida de un río, fallas, etc.)

- El proyecto GIS tratará con clientes y sucursales de Estados Unidos. Por lo tanto, el equipo de administración espacial decide:
  - Utilizar sistemas de coordenadas que definan con precisión las ubicaciones en las zonas de Estados Unidos en las que Safe Harbor realiza operaciones comerciales.
  - Utilizar el geocodificador por omisión, porque está diseñado para geocodificar direcciones de Estados Unidos.
- El equipo de administración espacial decide qué datos se necesitan para cumplir con los objetivos del proyecto y qué tablas contendrán dichos datos.

2. Mediante el Centro de control, el equipo de administración espacial crea dos sistemas de referencias espaciales. Uno determina el modo en que las coordenadas que definen las ubicaciones de las sucursales se deben convertir en elementos de datos que DB2 Spatial Extender pueda almacenar. El otro determina el modo en que las coordenadas que definen las ubicaciones de las residencias de los clientes se deben convertir en elementos de datos que DB2 Spatial Extender pueda almacenar.
3. Mediante el Centro de control, el equipo de administración espacial define columnas que contengan datos espaciales, las registra como capas y habilita un geocodificador para que las mantenga de forma automática:
   - El equipo añade una columna LOCATION a la tabla CUSTOMERS. La tabla ya contiene las direcciones de los clientes. El geocodificador por omisión las convertirá en datos espaciales y cargará estos datos en la columna LOCATION.
   - El equipo genera una tabla OFFICES para que contenga los datos que ahora se almacenan en el otro GIS. Estos datos incluyen las direcciones de las sucursales de Safe Harbor, datos espaciales que el geocodificador ha obtenido a partir de estas direcciones y datos espaciales que definen una zona dentro de un radio de cinco millas alrededor de cada sucursal. Los datos generados por el geocodificador irán en una columna LOCATION. Los datos que definen las zonas irán en una columna SALES_AREA.
   - El equipo registra las dos columnas LOCATION y las columnas SALES_AREA como capas.
   - El equipo habilita el geocodificador por omisión para que mantenga de forma automática las dos columnas LOCATION.

4. El equipo de administración espacial llena la columna LOCATION de la tabla CUSTOMER, toda la tabla OFFICES y una nueva tabla HAZARD_ZONES:
   - El equipo utiliza el Centro de control para llenar la columna LOCATION de la tabla CUSTOMER:
     a. El equipo indica al geocodificador que inserte datos espaciales correspondientes a una dirección en la columna LOCATION únicamente bajo la siguiente condición: la coincidencia entre la dirección y su equivalente en los registros de la Oficina de censo de Estados Unidos debe ser del 100 por ciento. (Con DB2 Spatial Extender se proporciona un archivo de direcciones suministrado por la Oficina de censo. Para que el geocodificador pueda convertir una dirección en datos fuente en datos espaciales, el geocodificador debe intentar hacer coincidir esta dirección con su equivalente en el archivo. Los usuarios especifican el grado de precisión de la coincidencia para que los datos espaciales se coloquen en una tabla. Este porcentaje se denomina una *precisión*.)
     b. El equipo ejecuta el geocodificador en modalidad de proceso por lotes, de modo que pueda geocodificar todas las direcciones de la tabla en una operación. Ante la sorpresa del equipo, el geocodificador rechaza aproximadamente una de cada diez direcciones.
     c. El equipo supone que los rechazos corresponden a las nuevas direcciones que no tienen coincidencias exactas en los registros de la Oficina de censo. Para resolver el problema, el equipo reduce la precisión a 85.
     d. El equipo vuelve a ejecutar el geocodificador en modalidad de proceso por lotes. La tasa de direcciones rechazadas cae a un nivel aceptable.
   - Mediante el programa de utilidad que se suministra por el GIS que va por separado, el equipo carga los datos de sucursales en un archivo. Luego el equipo utiliza el Centro de control para importar estos datos del archivo a la nueva tabla OFFICES.
   - Mediante el Centro de control, el equipo crea una tabla HAZARD ZONES, registra sus columnas espaciales como capas e importa datos en la misma. Los datos proceden de un archivo suministrado por un proveedor de correlaciones.
5. Mediante el Centro de control, el equipo de administración espacial facilita acceso a las nuevas capas:
   - El equipo crea índices para las mismas.
   - El equipo crea una vista que une columnas procedentes de las tablas CUSTOMERS y HAZARD ZONES. Luego el equipo registra las columnas espaciales de la vista como capas.

6. El equipo de análisis espacial ejecuta consultas para obtener información que ayudará a cumplir con los objetivos originales: determinar dónde establecer nuevas sucursales y ajustar las primas en función de la proximidad de los clientes a áreas peligrosas.

Capítulo 2. Instalación de DB2 Spatial Extender

Este capítulo contiene instrucciones para instalar DB2 Spatial Extender. Se tratan los siguientes temas:

- "Configuración de DB2 Spatial Extender"
- "Requisitos del sistema"
- "Instalación de DB2 Spatial Extender" en la página 19
- "Verificación de la instalación" en la página 20
- "Consideraciones posteriores a la instalación" en la página 21
- "Qué hacer a continuación" en la página 22

Configuración de DB2 Spatial Extender

Un sistema DB2 Spatial Extender consta de DB2 Universal Database, DB2 Spatial Extender y un geoexaminador (por ejemplo, ArcExplorer). Normalmente, hay una base de datos habilitada para operaciones espaciales en el servidor. Puede utilizar aplicaciones cliente para acceder a datos espaciales a través de consultas espaciales y procedimientos almacenados de DB2 Spatial Extender. Además, puede ver datos espaciales con un geoexaminador.

La Figura 5 ilustra la arquitectura de DB2 Spatial Extender.

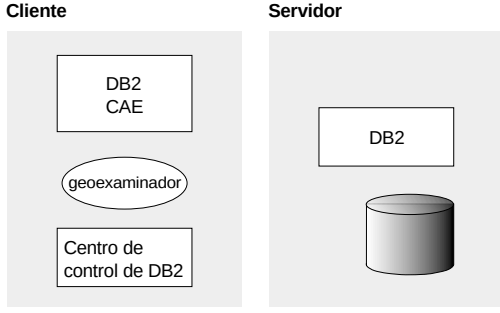


Figura 5. Configuración de cliente-servidor

Requisitos del sistema

Esta sección explica los requisitos de software y hardware para DB2 Spatial Extender.

Sistemas operativos soportados

DB2 Spatial Extender se puede instalar en los siguientes sistemas operativos:

- AIX 4.2 o posterior
- Windows NT 4.0 o posterior con Service Pack 5

Software de base de datos necesario

Antes de instalar DB2 Spatial Extender, debe tener software de DB2 instalado y configurado en el sistema. La Tabla 1 contiene los requisitos de software para el componente cliente de DB2 Spatial Extender y el componente servidor de DB2 Spatial Extender.

Tabla 1. Requisitos mínimos de software

Componente	Software
Cliente	DB2 Administration Client, Versión 7.1[1]
Servidor	Uno de los siguientes: • DB2 Universal Database Enterprise Edition, Versión 7.1 • DB2 Universal Database Enterprise – Extended Edition, Versión 7.1[2]
	Notas: 1. Si *no* tiene intención de utilizar el Centro de control de DB2, un geoexaminador para acceder a datos espaciales ni el programa de ejemplo de DB2 Spatial Extender, puede utilizar una versión inferior de DB2 Administration Client. 2. Aunque puede utilizar DB2 Spatial Extender con DB2 Universal Database Enterprise - Extended Edition, el índice espacial no se puede particionar entre varios nodos como en el entorno de proceso masivo en paralelo (MPP).

Requisitos de espacio en disco

La Tabla 2 contiene los requisitos de espacio en disco recomendados para DB2 Spatial Extender.

Tabla 2. Requisitos de espacio en disco

Componente de DB2 Spatial Extender	Espacio en disco
Biblioteca de servidor de DB2 Spatial Extender (incluye biblioteca de servidor de DB2 Spatial Extender, datos de referencia del geocodificador y documentación)	600 MB
Soporte de cliente de DB2 Spatial Extender (incluye los datos del programa de ejemplo)	15 MB

Instalación de DB2 Spatial Extender

Esta sección contiene la información que necesita para instalar DB2 Spatial Extender en los sistemas operativos Windows NT y AIX.

Antes de empezar

Si aún no lo ha hecho, instale DB2 Administration Client (herramientas de administración que incluyen el Centro de control y Run-Time Client) en la estación de trabajo cliente e instale DB2 Universal Database Enterprise Edition o DB2 Universal Database Enterprise - Extended Edition. Encontrará instrucciones para hacerlo en el manual *Guía rápida de iniciación* correspondiente.

Instalación de DB2 Spatial Extender en sistemas Windows NT

Para instalar DB2 Spatial Extender en un sistema Windows NT:

1. Conéctese al sistema con un nombre de usuario que tenga los permisos de administración necesarios.
2. Concluya cualquier otro programa.
3. Inserte el CD-ROM en la unidad. Se abrirá el área de ejecución de instalación.
4. Opcional: Pulse **Notas de release** para comprobar en la Notas de release de DB2 Spatial Extender si ha habido cambios en el proceso de instalación y luego vuelva al área de ejecución de DB2 Spatial Extender.
5. Pulse **Instalar**.
6. Responda a las solicitudes del programa de instalación. Dispone de ayuda en línea que le guiará por los pasos a seguir. Para invocar la ayuda en línea, pulse **Ayuda** o pulse la tecla F1.

Una vez finalizada la instalación, DB2 Spatial Extender queda instalado bajo el directorio %DB2PATH% (por ejemplo, c:\sqllib).

Instalación de DB2 Spatial Extender en sistemas AIX

Para instalar DB2 Spatial Extender en un sistema AIX

1. Inicie una sesión como root.
2. Inserte el CD-ROM en la unidad.
3. Monte el CD-ROM en su sistema AIX. Para obtener información sobre cómo montar un CD-ROM, consulte el manual *IBM DB2 Universal Database para UNIX Guía rápida de iniciación*.
4. Cambie al directorio en el que está montado el CD-ROM entrando el siguiente mandato:

   ```
   cd /cdrom
   ```

   donde *cdrom* es el punto de montaje de la unidad de CD-ROM en AIX.
5. Entre el mandato **db2setup** para iniciar el programa de instalación de DB2. Se abrirá la ventana Instalar DB2 Spatial Extender.

Nota: El programa de instalación de DB2 comenzará a funcionar con lentitud porque explora el sistema en busca de información.

6. Desde la lista de productos de la ventana Instalar DB2 Spatial Extender, seleccione los productos que desee instalar y pulse **Bien**.

   Para obtener información o ayuda para instalar DB2 Spatial Extender, pulse **Ayuda**.

Una vez terminada la instalación, DB2 Spatial Extender queda instalado en el directorio /usr/lpp/db2_07_01.

Verificación de la instalación

Después de instalar DB2 Spatial Extender, puede verificar su instalación utilizando el programa de ejemplo de DB2 Spatial Extender. Para poder ejecutar el programa de ejemplo, debe crear una base de datos de ejemplo y convertirla en ejecutable.

Nota: Asegúrese de utilizar el compilador especificado en el archivo makefile de DB2 Spatial Extender.

Para compilar y ejecutar el programa de ejemplo para Windows NT:

1. Inicie una sesión con el ID de usuario que tiene privilegios de administrador.
2. Desde un indicador de mandatos, entre **db2sampl** para crear la base de datos SAMPLE de DB2.
3. Desde un indicador de línea de mandatos, entre el siguiente mandato:

   ```
   cd %DB2PATH%\samples\spatial
   ```

Nota: Para poder llevar a cabo el paso 3 y continuar verificando la instalación, debe ser propietario de la instancia por omisión de DB2 (DB2–DB2).

4. Entre **make rungsedemo**.
5. Entre **rungsedemo.exe**.
6. Compruebe los mensajes de error y de finalización que se muestran a medida que se ejecuta el programa.

Para compilar y ejecutar el programa de ejemplo para AIX:

1. Inicie una sesión como root.
2. Cree o actualice una instancia de DB2.
3. Desde un indicador de mandatos, entre **db2sampl** para crear la base de datos SAMPLE de DB2.
4. Desde un indicador de línea de mandatos, entre el siguiente mandato:

   ```
   cd $DB2INSTANCE/sqllib/samples/spatial
   ```

Nota: Para poder llevar a cabo el paso 4 en la página 20 y continuar verificando la instalación, debe ser propietario de la instancia de DB2 que ha creado o actualizado.

5. Entre **make rungsedemo**.
6. Entre **rungsedemo**.
7. Compruebe los mensajes de error y de finalización que se muestran a medida que se ejecuta el programa.

Para obtener información sobre los programas de ejemplo, consulte el "Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 59.

Consideraciones posteriores a la instalación

Después de instalar DB2 Spatial Extender satisfactoriamente, debe tener en cuenta la posibilidad de:

- Bajar ArcExplorer
- Ejecutar el programa de utilidad de actualización de instancias de DB2

Cómo bajar ArcExplorer

IBM distribuye ArcExplorer Java 3.0 como un programa de ejemplo, que también puede obtener del sitio Web de ESRI, en http://www.esri.com.

Para obtener más información sobre cómo instalar y utilizar ArcExplorer, consulte el manual *Using ArcExplorer*, que también está disponible en el sitio Web de ESRI.

ArcExplorer necesita Java® 2 Runtime Environment (Standard Edition o Enterprise Edition) V1.2.2, disponible de forma gratuita en el sitio Web de Sun, en la dirección http://java.sun.com.

Importante: DB2 Universal Database V7.1 se suministra con IBM JDK 1.1.8. Cuando instale JRE 1.2.2 para ArcExplorer, colóquelo en un directorio distinto al de DB2. Recuerde definir correctamente la variable de entorno CLASSPATH.

Ejecución del programa de utilidad de actualización de instancias de DB2 (db2iupdt)

El programa de utilidad db2iupdt actualiza una instancia de DB2 especificada para:

- Permitir que la instancia adquiera una nueva configuración del sistema.
- Permitir que la instancia acceda a una función asociada a la instalación o eliminación de determinadas opciones de productos.

En AIX, este programa de utilidad se encuentra en /usr/lpp/db2_07_01. Si necesita ayuda, escriba `db2iupdt -h` en la línea de mandatos para abrir el

menú de ayuda. En el sistema operativo Windows NT, db2iupdt se encuentra en el directorio \sqllib\bin. Cambie a dicho directorio para escribir el mandato. Para ver una descripción completa de este mandato, consulte el manual *IBM DB2 Universal Database Consulta de mandatos*.

Qué hacer a continuación

Una vez instalado DB2 Spatial Extender, puede utilizar el Centro de control de DB2 para configurar el entorno GIS y comenzar a trabajar con información espacial.

Para invocar DB2 Spatial Extender desde el Centro de control:

1. Desde la ventana Centro de control, amplíe el árbol de objetos hasta encontrar la carpeta **Bases de datos** bajo el servidor en el que desea ejecutar DB2 Spatial Extender.
2. Pulse la carpeta **Bases de datos**. Se muestran las bases de datos en el panel de contenido en la parte derecha de la ventana.
3. Pulse con el botón derecho del ratón la base de datos con la que desea trabajar y pulse en el menú emergente la operación espacial que desea realizar.

Para obtener más información sobre cómo utilizar DB2 Spatial Extender desde el Centro de control, consulte:

- "Capítulo 3. Configuración de recursos" en la página 23
- "Capítulo 4. Cómo definir columnas espaciales, registrarlas como capas y habilitar un geocodificador para que las mantenga" en la página 33
- "Capítulo 5. Cómo llenar columnas espaciales" en la página 41
- "Capítulo 6. Creación de índices espaciales" en la página 53

Capítulo 3. Configuración de recursos

Después de instalar DB2 Spatial Extender, ya está listo para suministrar a su base de datos los recursos que necesita para crear columnas espaciales y manipular datos espaciales. Este capítulo resume estos recursos y describe dos de las tareas mediante las cuales los deja disponibles: habilitar la base de datos para operaciones espaciales y crear sistemas de referencias espaciales.

Inventario de recursos

Los recursos que necesita para crear columnas espaciales y para manipular datos espaciales incluyen:

- *Datos de referencia*: direcciones que DB2 Spatial Extender comprueba para verificar las direcciones que desea geocodificar
- Recursos que habilitan una base de datos para operaciones espaciales: procedimientos almacenados, funciones espaciales y otros
- Geocodificadores, distintos del geocodificador por omisión, que suministran usuarios y proveedores
- Sistemas de referencias espaciales

Esta sección trata sobre los datos de referencia y los recursos que habilitan una base de datos para operaciones espaciales. Para obtener información sobre geocodificadores distintos del geocodificador por omisión, consulte el tema “Acerca de la geocodificación” en la página 41. Para obtener información sobre sistemas de referencias espaciales, consulte el tema “Acerca de los sistemas de coordenadas y de referencias espaciales” en la página 25.

Datos de referencia

Los *datos de referencia* consisten en las direcciones más recientes de Estados Unidos que ha recopilado la Oficina de censo de Estados Unidos. Para que el geocodificador por omisión pueda convertir una dirección de la base de datos en coordenadas, debe comparar la dirección o parte de la misma con una dirección de los datos de referencia.

Los datos de referencia están disponibles desde el momento que instala DB2 Spatial Extender. Para ver la cantidad de espacio en disco que necesitan estos datos, consulte el tema “Requisitos de espacio en disco” en la página 18. Para comprobar en AIX que los datos se han cargado correctamente, búsquelos en el directorio $DB2INSTANCE/sqllib/gse/refdata/. Para comprobar en Windows NT que los datos se han cargado correctamente, búsquelos en el directorio %DB2PATH%\gse\refdata\.

Recursos que habilitan una base de datos para operaciones espaciales

La primera tarea a llevar a cabo después de instalar DB2 Spatial Extender consiste en habilitar la base de datos para operaciones espaciales. Esto implica iniciar una acción que haga que DB2 Spatial Extender cargue la base de datos con los siguientes recursos:

- Procedimientos almacenados. Cuando solicita una acción del Centro de control, DB2 Spatial Extender invoca uno de estos procedimientos almacenados para ejecutar la acción.
- Tipos de datos espaciales. Debe asignar un tipo de datos espaciales a cada columna de vista o tabla en la que se van a almacenar datos espaciales. Para obtener más información, consulte el apartado “Acerca de los tipos de datos espaciales” en la página 33.
- Vistas y tablas de catálogo de DB2 Spatial Extender. Ciertas operaciones dependen del catálogo de DB2 Spatial Extender. Por ejemplo, para que se pueda llenar una columna con un tipo de datos espaciales, se debe registrar en el catálogo como una capa. Para obtener información sobre capas, consulte el tema “Desarrollo e implantación de un proyecto GIS” en la página 10.
- Un tipo de índice espacial. Le permite definir índices para capas.
- Funciones espaciales. Sirven para trabajar con datos espaciales de diversas formas; por ejemplo, para determinar las relaciones entre funciones geográficas y para generar más datos espaciales. Una de estas funciones es un geocodificador por omisión. Convierte direcciones de Estados Unidos en coordenadas y luego inserta estas coordenadas en columnas espaciales. Para obtener más información sobre funciones espaciales, consulte el “Capítulo 13. Geometrías y funciones espaciales asociadas” en la página 131 y el “Capítulo 14. Funciones espaciales para consultas SQL” en la página 169. Para obtener más información sobre el geocodificador por omisión, consulte el tema “Acerca de la geocodificación” en la página 41.
- Un esquema, denominado DB2GSE, que contiene los objetos que se acaban de listar.

Para obtener instrucciones sobre cómo utilizar el Centro de control para iniciar la carga de estos recursos, consulte el tema “Habilitación de una base de datos para operaciones espaciales” en la página 25. Para obtener instrucciones sobre cómo utilizar una rutina de un programa de aplicación para realizar la misma tarea, consulte el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59.

Habilitación de una base de datos para operaciones espaciales

Para saber qué autorización se necesita para habilitar una base de datos para operaciones espaciales, consulte el tema “Autorización” en la página 80.

Para habilitar una base de datos para operaciones espaciales desde el Centro de control:

1. Desde la ventana Centro de control, amplíe el árbol de objetos hasta encontrar la carpeta **Bases de datos** bajo el servidor en el que desea ejecutar DB2 Spatial Extender.
2. Pulse la carpeta **Bases de datos**. Se muestran las bases de datos en el panel de contenido en la parte derecha de la ventana.
3. Pulse con el botón derecho del ratón la base de datos que desee y pulse **Spatial Extender** —> **Habilitar** en el menú emergente. DB2 Spatial Extender suministra a la base de datos los recursos que le permiten crear columnas y datos espaciales y trabajar con los mismos.

Recuerde: Para poder habilitar una base de datos para operaciones espaciales, DB2 Spatial Extender debe estar instalado en el servidor en el que reside la base de datos.

Creación de un sistema de referencias espaciales

Esta sección describe la relación entre sistemas de referencias espaciales y sistemas de coordenadas y explica cómo crear un sistema de referencias espaciales desde el Centro de control.

Acerca de los sistemas de coordenadas y de referencias espaciales

Esta sección continúa la explicación sobre sistemas de coordenadas iniciada en el tema “La naturaleza de los datos espaciales” en la página 6. Luego explica con más detalle los sistemas de referencias espaciales definidos en el tema “Desarrollo e implantación de un proyecto GIS” en la página 10. También contiene instrucciones para determinar qué valores asignar a los parámetros del sistema de referencias espaciales.

Sistemas de coordenadas, coordenadas y medidas

Puede imaginarse un sistema de coordenadas como una cuadrícula imaginaria que cubre una determinada área geográfica. Por ejemplo, una cuadrícula que cubre la tierra, una cuadrícula que cubre un país o una cuadrícula que cubre una región de un estado. Cada función geográfica del área se sitúa en la intersección de una línea de cuadrícula este-oeste y una línea de cuadrícula norte-sur. Un valor, denominado *coordenada X*, indica dónde se sitúa la ubicación en la línea de cuadrícula este-oeste. Otro valor, denominado *coordenada Y*, indica dónde se sitúa la ubicación en la línea de cuadrícula norte-sur. Ambos valores hacen referencia a la ubicación con respecto al centro de la cuadrícula u *origen*.

Las coordenadas X e Y del origen son ambas cero. A partir del origen hacia el este, las coordenadas X son positivas; a partir del origen hacia el oeste, son negativas. Paralelamente, a partir del origen hacia el norte, las coordenadas Y son positivas; a partir del origen hacia el sur, son negativas. Para ver una ilustración de esta distribución, pensemos en el siguiente ejemplo generalizado: el sistema de coordenadas A incluye una cuadrícula que cubre una gran área metropolitana. Una coordenada X de 7 indicaría una posición que está siete unidades al este del origen de esta cuadrícula. Una coordenada X de -9,5 indicaría una posición que está nueve unidades y media al oeste del origen.

Cada elemento de datos de una columna espacial incluye (1) una coordenada X y una coordenada Y que definen la ubicación de una función geográfica o (2) varias coordenadas X e Y que definen las ubicaciones de partes de una función o que definen el área que cubre una función. También se pueden incluir otros dos tipos de valores: una *coordenada Z* y una *medida*. A diferencia de las coordenadas X e Y, las coordenadas Z y las medidas no se utilizan en DB2 Spatial Extender para definir ubicaciones o áreas. Simplemente contienen información que necesita una aplicación GIS. Una coordenada Z suele indicar la altura o profundidad de una función geográfica. Las coordenadas Z que están por encima del origen son positivas; las que están por debajo son negativas. Una medida es numérica; puede contener cualquier tipo de información. Por ejemplo, supongamos que está representando pozos de petróleo en su GIS. Si necesita aplicaciones para procesar valores que indican ID de puntos identificados correspondientes a datos sísmicos, puede almacenar estos valores como medidas.

Sistemas de referencias espaciales, desplazamientos y factores de escala

Tal como se indica en el tema “Sistemas de coordenadas, coordenadas y medidas” en la página 25, las coordenadas pueden ser negativas y se pueden expresar en decimales. Esto también se aplica a las medidas. Sin embargo, para reducir la carga general del almacenamiento, DB2 Spatial Extender almacena cada coordenada y medida como un entero no negativo (es decir, como un entero positivo o cero). Por lo tanto, las coordenadas y medidas reales negativas y decimales se deben convertir en enteros no negativos para que DB2 Spatial Extender las pueda guardar. Además, debe indicar a DB2 Spatial Extender cómo realizar la conversión. Para ello debe definir ciertos parámetros. Los valores de los parámetros que se utilizarán para convertir coordenadas y medidas dentro de un área geográfica se denominan de forma colectiva *sistema de referencias espaciales*.

Puede crear un sistema de referencias espaciales:

- Determinando las medidas y coordenadas negativas menores correspondientes a las funciones que está representando. (Cuanto más lejos

de cero esté un valor negativo, menor es. Una coordenada X de –10 es menor que una coordenada X de –5; una medida de –100 es menor que una medida de –50.)

- Especificando *factores de desplazamiento* (o *desplazamientos*, para abreviar): valores que, cuando se restan de medidas y coordenadas negativas, dan como resultado números no negativos.
- Especificando *factores de escala*: valores que, cuando se multiplican por medidas y coordenadas decimales, dan lugar a enteros cuya precisión es mayor o igual al de las coordenadas o medidas. Por ejemplo, supongamos que tenemos una coordenada con una precisión de cuatro: 92,77. Podría multiplicarla por un factor de escala de 100 para obtener un entero con una precisión de cuatro: 9277.

Determinación de las coordenadas y medidas negativas más bajas

Para poder definir parámetros para un sistema de referencias espaciales, tiene que determinar la coordenada X, la coordenada Y, la coordenada Z y la medida más bajas del área geográfica que contiene las funciones sobre las que desea información. Puede buscar estos valores contestando las siguientes preguntas:

- Entre las funciones que está representando, ¿hay alguna que quede al oeste del origen del sistema de coordenadas que está utilizando? Si es así, ¿qué coordenada X indica la ubicación de la esquina oeste de la función que queda más hacia el oeste? (La respuesta será la coordenada X negativa más baja que está manejando.) Por ejemplo, si está representando pozos de petróleo, y algunos quedan al oeste del origen, ¿qué coordenada X indica la ubicación del pozo de petróleo que queda más hacia el oeste?
- ¿Hay alguna función al sur del origen? Si es así, ¿qué coordenada Y indica la ubicación de la esquina sur de la función que queda más hacia el sur? (La respuesta será la coordenada Y negativa más baja que está manejando.) Por ejemplo, si está representando pozos de petróleo, y algunos quedan al sur del origen, ¿qué coordenada Y indica la ubicación del pozo de petróleo que queda más hacia el sur?
- Si va a utilizar coordenadas Z para definir profundidad, ¿qué función es la más profunda y qué coordenada Z representa el punto más bajo de esta función? (La respuesta será la coordenada Z negativa más baja que está manejando.)
- Si va a incluir medidas en sus datos espaciales ¿habrá alguna negativa? Si es así, ¿cuál es la medida negativa más baja?

Una vez determinadas las coordenadas y medidas negativas más bajas, sume a cada una de ellas una cantidad igual a entre el cinco y el diez por ciento de su valor. Por ejemplo, si la coordenada X negativa más baja es –100, podría sumarle –5. En este manual se denomina al número resultante un *valor aumentado*.

Especificación de factores de desplazamiento

A continuación, especifique qué factores de desplazamiento debe utilizar DB2 Spatial Extender para convertir coordenadas y medidas negativas en coordenadas y medidas no negativas:

- Después de decidir qué valor de X aumentada desea, especifique un desplazamiento que, cuando se reste de este valor, dé como resultado cero. DB2 Spatial Extender restará este número de todas las coordenadas X negativas para llegar a un valor positivo. DB2 Spatial Extender también restará este número de todas las demás coordenadas X.

  Por ejemplo, si el valor de X aumentada es –105, tiene que restarle –105 para conseguir un valor igual a 0. Luego DB2 Spatial Extender restará –105 de todas las coordenadas X asociadas a las funciones que está representando. Puesto que ninguna de estas coordenadas es mayor que –100, todos los valores resultantes de la resta serán positivos.
- Paralelamente, especifique desplazamientos que den como resultado 0 cuando se resten del valor de Y aumentada, del valor de Z aumentada y de la medida aumentada.

El desplazamiento restado de coordenadas X se denomina *X falsa*. Los desplazamientos restados de coordenadas Y, de coordenadas Z y de medidas se denominan *Y falsa*, *Z falsa* y *M falsa* respectivamente. Para ver instrucciones sobre cómo especificar estos parámetros desde el Centro de control, consulte el tema “Creación de un sistema de referencias espaciales desde el Centro de control” en la página 29.

Especificación de factores de escala

A continuación, especifique qué factores de escala debe utilizar DB2 Spatial Extender para convertir medidas y coordenadas decimales en enteros:

- Especifique un factor de escala que, cuando se multiplique por una coordenada X decimal o por una coordenada Y decimal, dé como resultado un entero de 32 bits. Se recomienda que este factor de escala sea un factor de 10: 10 a la primera potencia (10), 10 a la segunda potencia (100), 10 a la tercera potencia (1000), o, si es necesario, un factor mayor. Para decidir qué factor de 10 debe ser el factor de escala:
  1. Determine qué coordenadas X e Y son números decimales o tienen bastante probabilidad de serlo. Por ejemplo, supongamos que determina que tres de las diversas coordenadas X e Y que va a manejar son números decimales: 1,23, 5,1235 y 6,789.
  2. Tome la coordenada decimal que tiene la precisión decimal más larga. Luego determine por qué factor de diez se puede multiplicar esta coordenada para obtener un entero de igual precisión. Para ilustrar: de las tres coordenadas decimales del ejemplo actual, 5,1235 tiene la precisión decimal más larga. Si la multiplicamos por diez a la cuarta potencia (10000), obtendremos un entero, 51235.

3. Determine qué entero generado por la multiplicación que se acaba de describir es demasiado largo para almacenarlo como un elemento de datos de 32 bits. 51235 no es demasiado largo. Pero supongamos que, además de 1,23, 5,11235 y 6,789, su rango de coordenadas X e Y incluye un cuarto valor decimal, 10006,789876. Puesto que la precisión decimal de esta coordenada es más larga que las de las otras tres, tendría que multiplicar *esta* coordenada—no 5,1235—por un factor de 10. Para convertirla en un entero, la tendría que multiplicar por 10 a la sexta potencia (1000000). Pero el valor resultante, 10006789876, es demasiado largo para almacenarlo como un elemento de datos de 32 bits. Si DB2 Spatial Extender intentara almacenarlo, los resultados serían imprevisibles.

   Para evitar este problema, seleccione un factor de 10 que, multiplicado por la coordenada original, dé como resultado un número decimal que DB2 Spatial Extender pueda truncar en un entero que pueda almacenar, con la mínima pérdida de precisión. En este caso, podría seleccionar 10 a la cuarta potencia (10000). El resultado de multiplicar 10000 por 10006,789876 es 100067898,76. DB2 Spatial Extender truncaría este número a 100067898, reduciendo su precisión por una cantidad virtualmente insignificante.

- Si las funciones que está representando tienen coordenadas Z decimales, siga el siguiente procedimiento para determinar un factor de escala para estas coordenadas. Si las funciones están asociadas a medidas decimales, siga el mismo procedimiento para determinar un factor de escala para estas medidas.

El factor de escala para las coordenadas X e Y se denomina *unidad XY*. Los factores de escala para coordenadas Z y medidas se denominan *unidades Z* y *unidades M*, respectivamente. Para ver instrucciones sobre cómo especificar estos parámetros desde el Centro de control, consulte el tema “Creación de un sistema de referencias espaciales desde el Centro de control”.

Creación de un sistema de referencias espaciales desde el Centro de control

Esta sección contiene una visión general de los pasos a seguir para crear un sistema de referencias espaciales desde el Centro de control. La visión general va seguida de detalles sobre cómo llevar a cabo cada paso.

No se necesita ninguna autorización para llevar a cabo estos pasos.

Visión general para crear un sistema de referencias espaciales desde el Centro de control:

1. Abra la ventana Crear referencia espacial.
2. Indique qué sistema de coordenadas desea utilizar.

3. Especifique identificadores para el sistema de referencias espaciales que desea crear.
4. Determine qué rangos de coordenadas y medidas se aplican a las funciones geográficas sobre las que desea información.
5. Especifique valores que se puedan utilizar para convertir coordenadas y medidas negativas o decimales en elementos de datos que DB2 Spatial Extender pueda almacenar.
6. Indique a DB2 Spatial Extender que cree el sistema de referencias espaciales que desee.

Pasos detallados para crear un sistema de referencias espaciales desde el Centro de control:

1. Abra la ventana Crear referencia espacial.
   a. Desde la ventana Centro de control, amplíe el árbol de objetos hasta encontrar la carpeta **Bases de datos** bajo el servidor en el que desea ejecutar DB2 Spatial Extender.
   b. Pulse la carpeta **Bases de datos**. Se muestran las bases de datos en el panel de contenido en la parte derecha de la ventana.
   c. Pulse con el botón derecho del ratón la base de datos que ha habilitado para datos espaciales y pulse **Spatial Extender** —> **Referencias espaciales** en el menú emergente. Se abrirá la ventana Referencias espaciales.
   d. En la ventana Referencias espaciales, pulse **Crear**. Se abrirá la ventana Crear referencia espacial.
2. En la ventana Crear referencia espacial, utilice el campo **Sistema de coordenadas** para indicar qué sistema de coordenadas desea utilizar.
3. Especifique identificadores para el sistema de referencias espaciales que desea crear.
   - En el campo **Nombre**, escriba un nombre de entre 1 y 64 caracteres para el sistema.

     Restricción: No especifique el nombre de otro sistema de referencias espaciales. En la base de datos no puede haber dos sistemas de referencias espaciales con el mismo nombre.
   - En el campo ID, escriba un identificador numérico. Debe ser un entero.

     Restricción: No especifique el ID de otro sistema de referencias espaciales. En la base de datos no puede haber dos sistemas de referencias espaciales con el mismo ID.
4. En un medio externo al Centro de control—como por ejemplo un papel o una pizarra—determine las coordenadas y medidas negativas más bajas que se aplican a las funciones geográficas que está representando. Para

obtener instrucciones sobre cómo hacerlo, consulte el tema "Determinación de las coordenadas y medidas negativas más bajas" en la página 27.

5. En la ventana Crear referencia espacial, especifique valores para convertir coordenadas y medidas negativas o decimales en elementos de datos que DB2 Spatial Extender soporta—es decir, enteros no negativos de 32 bits.
   a. Especifique valores para convertir coordenadas X negativas o decimales en enteros no negativos:
      - En la columna **Desplazamiento**, en el campo más cercano a la **X**, especifique una X falsa:
        – Si alguno de los valores dentro del rango de coordenadas X identificado en el paso 4 en la página 30 es negativo, escriba una X falsa que, cuando se reste de la coordenada negativa más baja, dé como resultado un número positivo. Para obtener instrucciones, consulte el tema "Especificación de factores de desplazamiento" en la página 28.
        – Si todas las coordenadas X son no negativas, escriba una X falsa de 0.
      - En la columna **Factor de escala**, especifique una unidad XY en el campo que hay más a la derecha de la **X**. Esta unidad XY debe ser una que, multiplicada por cualquier coordenada X decimal o coordenada Y decimal, dé como resultado un número entero que se pueda almacenar como un elemento de datos de 32 bits, con la mínima pérdida de precisión. Para obtener instrucciones, consulte el tema "Especificación de factores de escala" en la página 28.

        Después de especificar la unidad XY en el campo que hay más a la derecha de la **X**, aparecerá también en el campo que hay más a la derecha de la **Y**.
   b. Especifique una Y falsa que permita a DB2 Spatial Extender convertir coordenadas Y negativas en valores positivos. Esto se especifica en la columna **Desplazamiento**, en el campo más cercano a la **Y**:
      - Si alguno de los valores dentro del rango de coordenadas Y identificado en el paso 4 en la página 30 es negativo, escriba una Y falsa que, cuando se reste de la coordenada negativa más baja, dé como resultado un número positivo. Para obtener instrucciones, consulte el tema "Especificación de factores de desplazamiento" en la página 28.
      - Si todas las coordenadas Y son positivas, escriba una Y falsa de 0.
   c. Si va a incluir coordenadas Z en sus datos espaciales, especifique valores para convertir coordenadas Z decimales o negativas en enteros no negativos:
      - En la columna **Desplazamiento**, en el campo más cercano a la **Z**, escriba una Z falsa:

- Si alguno de los valores dentro del rango de coordenadas Z identificado en el paso 4 en la página 30 es negativo, escriba una Z falsa que, cuando se reste de la coordenada negativa más baja, dé como resultado un número positivo. Para obtener instrucciones, consulte el tema "Especificación de factores de desplazamiento" en la página 28.
- Si todas las coordenadas Z son no negativas, escriba una Z falsa de 0.

• En la columna **Factor de escala**, especifique una unidad Z en el campo que hay más a la derecha de la **Z**. Esta unidad Z debe ser una que, multiplicada por cualquier coordenada Z decimal, dé como resultado un número entero que se pueda almacenar como un elemento de datos de 32 bits, con la mínima pérdida de precisión. Para obtener instrucciones, consulte el tema "Especificación de factores de escala" en la página 28.

d. Si va a incluir medidas en sus datos espaciales, especifique valores para convertir medidas decimales o negativas en enteros positivos:

• En la columna **Desplazamiento**, en el campo más cercano a la etiqueta **Lineal**, escriba una M falsa:

- Si alguno de los valores dentro del rango de medidas identificado en el paso 4 en la página 30 es negativo, escriba una M falsa que, restada de la medida negativa menor, dé como resultado un número positivo. Para obtener instrucciones, consulte el tema "Especificación de factores de desplazamiento" en la página 28.
- Si todas las medidas son positivas, escriba una M falsa de 0.

• En la columna **Factor de escala**, especifique una unidad M en el campo que hay más a la derecha de la etiqueta **Lineal**. Esta unidad M debe ser una que, multiplicada por cualquier medida decimal, dé como resultado un número entero que se pueda almacenar como un elemento de datos de 32 bits, con la mínima pérdida de precisión. Para obtener instrucciones, consulte el tema "Especificación de factores de escala" en la página 28.

6. Pulse **Bien** para crear el sistema de referencias espaciales que desea.

Capítulo 4. Cómo definir columnas espaciales, registrarlas como capas y habilitar un geocodificador para que las mantenga

Una vez definidos los recursos para su GID de DB2 Spatial Extender, ya está listo para crear objetos que contendrán datos espaciales. Por ejemplo, si necesita nuevas tablas para que contengan datos espaciales, las puede definir asignando tipos de datos espaciales a las columnas en las que desea colocar los datos. Si necesita añadir columnas espaciales a tablas existentes, también puede hacerlo.

Cuando suministra a una tabla nueva o existente una columna espacial, tiene que registrar dicha columna como una capa. Además, si tiene intención de que un geocodificador llene la columna, puede, en el momento de registrar la columna como una capa, habilitar el geocodificador para que la mantenga de forma automática. Esta habilitación se produce del siguiente modo: DB2 Spatial Extender define activadores que están codificados para invocar el geocodificador siempre que la columna o columnas de atributo correspondientes a la columna espacial reciben datos nuevos o actualizados. Cuando se le invoca, el geocodificador convierte los datos nuevos o actualizados en datos espaciales y los coloca en la columna espacial.

Después de definir una columna espacial para una tabla puede, si lo desea, crear una columna de vista sobre esta columna de tabla. Debe registrar la columna de vista como una capa después de registrar la columna de la tabla como una capa.

Este capítulo describe la naturaleza y el uso de los tipos de datos que puede asignar a una columna espacial. A continuación, el capítulo explica cómo utilizar el Centro de control para definir una columna espacial para una tabla, para registrar esta columna como una capa y para habilitar un geocodificador para que la mantenga. Finalmente, el capítulo explica cómo utilizar el Centro de control para registrar una columna de vista como una capa.

Acerca de los tipos de datos espaciales

Esta sección contiene una introducción a los tipos de datos necesarios para las columnas espaciales y ofrece directrices para seleccionar cuál debe ser el tipo de datos de una columna espacial.

Cuando habilita una base de datos para operaciones espaciales, DB2 Spatial Extender suministra a la base de datos una jerarquía de tipos de datos

estructurados. La Figura 6 presenta esta jerarquía. En esta figura, los tipos con los que se pueden crear instancias tienen fondo blanco; los tipos con los que no se pueden crear instancias tienen fondo gris.

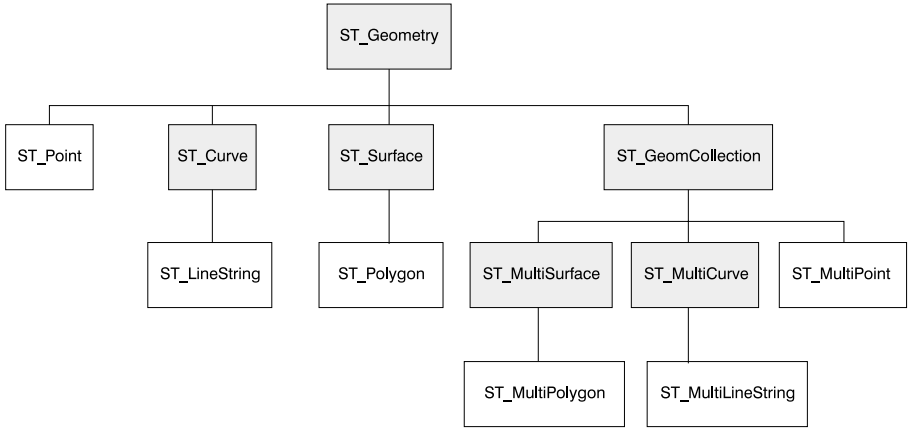


Figura 6. Jerarquía de tipos de datos espaciales. Con los tipos de datos que aparecen en recuadros blancos se pueden crear instancias. Con los tipos de datos que aparecen en recuadros grises no se pueden crear instancias.

La jerarquía de la Figura 6 incluye:

- Tipos de datos para funciones geográficas que se pueden percibir como si formaran una sola unidad; por ejemplo, residencias individuales y lagos aislados.
- Tipos de datos para funciones geográficas que constan de varias unidades o componentes; por ejemplo, sistemas de autopistas y cadenas de montañas.
- Un tipo de datos para funciones geográficas de todo tipo.

Tipos de datos para funciones de una sola unidad

Utilice ST_Point, ST_Linestring y ST_Polygon para almacenar coordenadas que definen el espacio ocupado por funciones que se pueden percibir como si formaran una sola unidad:

- Utilice ST_Point cuando desee indicar el punto del espacio ocupado por una función geográfica diferenciada. La función puede ser muy pequeña, como un pozo de agua; muy grande, como una ciudad; o de tamaño intermedio, como un complejo de edificios o un parque. En cada caso, el punto en el espacio se puede localizar en la intersección de una línea de coordenadas este-oeste (por ejemplo, un paralelo) y una línea de coordenadas norte-sur (por ejemplo, un meridiano). Un elemento de datos ST_Point incluye valores—una coordenada X y una coordenada Y—que definen esta intersección. La coordenada X indica el lugar donde se encuentra la intersección en la línea este-oeste; la coordenada Y indica el lugar donde se encuentra la intersección en la línea norte-sur.

- Utilice ST_Linestring para coordenadas que definen el espacio ocupado por funciones lineales; por ejemplo, calles, canales y conductos.
- Utilice ST_Polygon cuando desee indicar la extensión de espacio cubierto por una función de varios lados; por ejemplo, un distrito de prestaciones sociales, un bosque o un hábitat de vida salvaje. Un elemento de datos ST_Polygon consiste en las coordenadas que definen el perímetro de una función de este tipo.

  En algunos casos, se puede utilizar ST_Polygon y ST_Point para la misma función. Por ejemplo, supongamos que necesita información espacial sobre varios complejos de apartamentos. Si desea representar el punto del espacio en el que se encuentra cada complejo, utilizaría ST_Point para almacenar las coordenadas X e Y que definen cada punto. Por otro lado, si desea representar el área que cubre cada complejo, utilizaría ST_Polygon para almacenar las coordenadas que definen el perímetro de cada área.

Tipos de datos para funciones de varias unidades

Utilice ST_MultiPoint, ST_MultiLineString y ST_MultiPolygon para almacenar coordenadas que definen espacios ocupados por funciones formadas por varias unidades:

- Utilice ST_MultiPoint si desea representar funciones formadas por unidades diferenciadas y desea indicar el punto en el espacio ocupado por cada componente. Un elemento de datos ST_MultiPoint incluye los pares de coordenadas X e Y que definen la ubicación de cada componente de este tipo de función. Por ejemplo, pensemos en una tabla cuyas filas representan cadenas de islas y cuyas columnas incluyen una columna ST_MultiPoint. Cada elemento de datos de esta columna incluye pares de coordenadas X e Y que definen las ubicaciones de las islas en cada cadena.
- Utilice ST_MultiLineString cuando desee representar funciones formadas por unidades lineales y desee información sobre el espacio ocupado por cada unidad. Un elemento de datos ST_MultiLineString consta de las coordenadas que definen este tipo de espacios. Por ejemplo, pensemos en una tabla cuyas filas representan sistemas de ríos y cuyas columnas incluyen una columna ST_MultiLineString. Cada elemento de datos de esta columna incluye las series de coordenadas que definen los recorridos de los ríos de cada sistema.
- Utilice ST_MultiPolygon cuando desee representar funciones formadas por unidades de varios lados y desee información sobre el espacio ocupado por cada unidad. Por ejemplo, pensemos en una tabla cuyas filas representan países de Oriente Medio y cuyas columnas incluyen una columna ST_MultiPolygon. Esta columna contiene información sobre tierras de labranza. En concreto, cada elemento de datos de la columna incluye las series de coordenadas que definen los perímetros de las tierras de labranza de un determinado país.

Un tipo de datos para todas las funciones

Puede utilizar ST_Geometry cuando no esté seguro de qué otro tipo de datos debe utilizar. Puesto que ST_Geometry es la raíz de la jerarquía a la que pertenecen los otros tipos de datos, una columna ST_Geometry puede almacenar cualquiera de los valores (o todos ellos) que se pueden almacenar en columnas a las que están asignados los demás tipos de datos.

Atención: El geocodificador por omisión puede convertir direcciones en elementos de datos ST_Point o ST_Geometry. Por lo tanto, si tiene intención de utilizar este geocodificador para llenar una columna espacial, debe asignar el tipo de datos ST_Point o ST_Geometry a esta columna.

Cómo definir una columna espacial para una tabla, registrar esta columna como una capa y habilitar un geocodificador para que la mantenga

Esta sección contiene una visión general de los pasos a seguir para definir una columna espacial para una tabla, registrar esta columna como una capa y habilitar un geocodificador para que la mantenga. La visión general va seguida de detalles sobre cómo llevar a cabo cada paso.

Para saber qué autorización necesita para registrar una columna de la tabla como una capa, consulte el tema “Autorización” en la página 95. Para saber qué autorización necesita para habilitar un geocodificador para que mantenga esta columna, consulte el tema “Autorización” en la página 76.

Visión general de los pasos a seguir para definir una columna espacial para una tabla, registrar esta columna como una capa y habilitar un geocodificador para que la mantenga:

1. Si la columna espacial debe formar parte de una tabla nueva, cree dicha tabla.
2. Abra la ventana Crear capa espacial.
3. Añada una columna espacial a una tabla e indique que desea registrar esta columna como una capa, o bien indique que desea registrar una columna existente como una capa.
4. Indique qué sistema de referencias espaciales se debe utilizar para la capa.
5. Si la capa debe contener datos importados o datos generados a partir de otra columna espacial, indique a DB2 Spatial Extender que cree la capa.
6. Si la capa debe contener datos obtenidos a partir de datos de atributo:
   a. Especifique qué columna o columnas contienen estos datos de atributo.
   b. Indique que desea habilitar un geocodificador para que mantenga la capa.
   c. Indique a DB2 Spatial Extender que cree la capa.

Detalles paso a paso para definir una columna espacial para una tabla, registrar esta columna como una capa y habilitar un geocodificador para que la mantenga:

1. Si la columna espacial debe formar parte de una tabla nueva, cree dicha tabla:
   - Utilice la interfaz que desee (por ejemplo, el Centro de control o el procesador de línea de mandatos) para crear la tabla.
   - Si tiene intención de utilizar un geocodificador, incluya entre una y diez columnas sobre las que vaya a funcionar el geocodificador. Un geocodificador no puede tomar más de diez columnas de datos como entrada.
   - Incluya la columna espacial que vaya a registrar como una capa o defina esta columna en el paso 3.

   Si desea utilizar una tabla existente, continúe en el paso siguiente.
2. Abra la ventana Crear capa espacial.
   a. Desde la ventana Centro de control, amplíe el árbol de objetos hasta encontrar la carpeta **Tablas** en la que están las tablas de la base de datos que utiliza para operaciones espaciales.
   b. Pulse la carpeta **Tablas**. Se mostrarán las tablas en el panel de contenido de la parte derecha de la ventana.
   c. Pulse con el botón derecho del ratón la tabla que desee y pulse **Spatial Extender** —> **Capas espaciales** en el menú emergente. Se abrirá la ventana Capas espaciales.
   d. En la ventana Capas espaciales, pulse **Crear**. Se abrirá la ventana Crear capa espacial.
3. En la ventana Crear capa espacial, añada una columna espacial a una tabla, e indique que desea registrar esta columna como una capa, o bien indique que desea registrar una columna existente como una capa.
   - Si desea añadir una columna espacial a una tabla y definir dicha columna como una capa:
     a. En el campo **Columna de capa**, escriba un nombre para la columna.
     b. En el campo **Tipo de columna**, seleccione o escriba el tipo de datos que desea que tenga la columna. Para ver los tipos de datos permitidos, consulte el tema “Acerca de los tipos de datos espaciales” en la página 33.
   - Si desea definir una columna existente como una capa, selecciónela en el campo **Columna de capa**.

     Restricción: No seleccione una columna que ya se haya definido como una capa.
4. En el campo **Nombre de referencia espacial**, especifique el nombre del sistema de referencias espaciales a utilizar para la capa.

5. Si desea que la capa contenga datos importados o datos generados a partir de otra columna espacial, pulse **Bien** para registrarla.
6. Si desea que la capa contenga datos obtenidos a partir de datos de atributo:
   a. Especifique qué columna o columnas contienen estos datos de atributo:
      1) Seleccione la columna o columnas en el recuadro **Columnas disponibles**. Puede seleccionar un máximo de diez columnas.
      2) Pulse el botón >, el botón >> o ambos para listar la columna o columnas seleccionadas en el recuadro **Columnas seleccionadas**.
   b. Si desea habilitar un geocodificador para que mantenga la capa:
      1) Seleccione el recuadro de selección **Habilitar geocodificador automático**.
      2) En el campo **Nombre**, seleccione el nombre del geocodificador que desea utilizar.
      3) En el campo **Nivel de precisión**, especifique, en términos de porcentaje, el grado en que los registros de entrada deben coincidir con los registros correspondientes de los datos de referencia para que se procesen. Este porcentaje se denomina una *precisión*. Por ejemplo, supongamos que el geocodificador lee un registro de entrada que contiene la dirección 557 Bailey, San Jose 94120. Si la precisión es 100 y la coincidencia entre esta dirección y su correspondiente en los datos de referencia no es del 100 por cien, el geocodificador la rechazará. Si la precisión es 75 y la coincidencia entre el registro y su correspondiente en los datos de referencia es de al menos el 75 por ciento, el geocodificador lo procesará.
      4) Si el geocodificador procede de un proveedor, utilice el recuadro **Propiedades** para especificar los parámetros de geocodificación suministrados por el proveedor que desee utilizar.
   c. Pulse **Bien** para registrar la columna seleccionada como una capa y, si lo ha solicitado, para habilitar el geocodificador para que mantenga la columna.

Cómo registrar una columna de vista como una capa

Para ver qué autorización necesita para registrar una columna de vista como una capa, consulte el tema “Autorización” en la página 95.

Para registrar una columna de vista como una capa:

1. Abra la ventana Crear capa espacial.
   a. Desde la ventana Centro de control, amplíe el árbol de objetos hasta encontrar la carpeta **Vistas** en la que están las vistas de la base de datos que utiliza para operaciones espaciales.
   b. Pulse la carpeta **Vistas**. Se mostrarán las vistas en el panel de contenido de la parte derecha de la ventana.
   c. Pulse con el botón derecho del ratón la vista que desee y pulse **Spatial Extender** —> **Capas espaciales** en el menú emergente. Se abrirá la ventana Capas espaciales.
   d. En la ventana Capas espaciales, pulse **Crear**. Se abrirá la ventana Crear capa espacial.
2. Utilice el recuadro **Columna de capa** para especificar la columna que desea registrar como una capa.
3. En el campo **Capa espacial implícita**, especifique el nombre de la columna de tabla en la que se basa la columna de vista seleccionada. Esta columna de tabla ya debe estar registrada como una capa.
4. Pulse **Bien** para registrar la columna de vista especificada como una capa.

Capítulo 5. Cómo llenar columnas espaciales

Después de registrar las columnas espaciales como capas, está listo para suministrarles datos espaciales. Tal como se indica en el tema “De dónde proceden los datos espaciales” en la página 6, hay varias formas de suministrar estos datos: utilizando una función, denominada geocodificador, para que obtenga los datos a partir de datos de atributo, utilizando otras funciones para obtener datos a partir de otros datos espaciales o importando los datos desde archivos. Este capítulo:

- Contiene información sobre la geocodificación y explica cómo utilizar el Centro de control para geocodificar datos de atributo en modalidad de proceso por lotes
- Explica cómo importar y exportar datos y cómo utilizar el Centro de control para importar datos a su GIS y exportarlos de su GIS

Para obtener más información sobre funciones que permiten obtener nuevos datos espaciales a partir de datos espaciales existentes, consulte el tema “Funciones que generan geometrías nuevas a partir de geometrías existentes” en la página 158.

Utilización de geocodificadores

Esta sección describe el proceso de geocodificar y explica cómo ejecutar un geocodificador en modalidad de proceso por lotes desde el Centro de control.

Acerca de la geocodificación

Esta sección explica las diferencias básicas entre geocodificadores y sus fuentes. También describe las dos modalidades en que se puede ejecutar un geocodificador y contiene una introducción sobre los factores a tener en cuenta cuando se tiene intención de utilizar un geocodificador.

Con DB2 Spatial Extender, puede:

- Utilizar el geocodificador por omisión que se suministra con DB2 Spatial Extender.
- Conectar geocodificadores desarrollados por otros proveedores.
- Conectar sus propios geocodificadores.

El geocodificador por omisión geocodifica direcciones de Estados Unidos y permite convertirlas en datos ST_Point o en datos ST_Geometry. Si tiene que almacenar datos de otros tipos de datos espaciales, puede conectar un geocodificador que genere dichos datos. Si necesita datos espaciales que representen lugares de fuera de Estados Unidos, o lugares que no tienen

dirección—por ejemplo, tierras de labranza que se distinguen por el contenido de la tierra—puede conectar un geocodificador que se ajuste a sus requisitos.

Para poder utilizar un geocodificador conectado, este debe estar registrado. Los usuarios y proveedores pueden registrar sus geocodificadores con el procedimiento almacenado db2gse.gse_register_gc. No se pueden registrar desde el Centro de control. Para obtener información sobre db2gse.gse_register_gc, consulte el tema “db2gse.gse_register_gc” en la página 93. Para obtener información general sobre cómo utilizar los procedimientos almacenados de DB2 Spatial Extender, consulte el “Capítulo 9. Procedimientos almacenados” en la página 69.

Un geocodificador se puede ejecutar en dos modalidades:

- En *modalidad de proceso por lotes* intenta, en una sola operación, convertir todos los datos fuente existentes correspondientes a una columna espacial en datos espaciales y llenar la columna con dichos datos. Puede iniciar esta operación desde la ventana Ejecutar geocodificador. También puede iniciarla en un programa de aplicación, codificando el programa para que llame al procedimiento almacenado db2gse.gse_run_gc.
- En *modalidad incremental*, un geocodificador convierte datos cuando estos se insertan o actualizan en una tabla, colocando los valores espaciales resultantes en una columna a fin de mantener la columna actualizada. Se activa mediante activadores de inserción y actualización que puede solicitar desde la ventana Crear capa espacial. También puede solicitarlos en un programa de aplicación, codificando el programa para que llame al procedimiento almacenado db2gse.gse_enable_autogc.

  La geocodificación incremental también recibe el nombre de *geocodificación automática*.

Cuando tenga intención de utilizar un geocodificador, tenga en cuenta los siguientes factores:

1. Cuando utilice el Centro de control, generalmente utilizará la ventana Crear capas espaciales antes de utilizar la ventana Ejecutar geocodificador. Esto significa que puede indicar a DB2 Spatial Extender que configure activadores para la geocodificación incremental antes de iniciar la geocodificación de proceso por lotes. Por lo tanto, es posible que la geocodificación incremental preceda a la geocodificación de proceso por lotes. Al procesar todos los datos fuente en modalidad de proceso por lotes, el geocodificador geocodificará los mismos datos utilizados en la modalidad incremental. Esta redundancia no causará duplicaciones (cuando los datos espaciales se producen dos veces, el segundo grupo de datos prevalece sobre el primero). Sin embargo, esto puede degradar el rendimiento. Un modo de evitarlo consiste en diferir la configuración de activadores hasta que termine la geocodificación de proceso por lotes.

2. Si los activadores están listos cuando el usuario esté preparado para geocodificar en modalidad de proceso por lotes, se recomienda desactivarlos hasta que termine la geocodificación de proceso por lotes. Puede desactivarlos desde la ventana Ejecutar geocodificador o en un programa de aplicación, codificando el programa para que llame al procedimiento almacenado db2gse.gse_disable_autogc. Si utiliza la ventana Ejecutar geocodificador, DB2 Spatial Extender los vuelve a activar automáticamente cuando termina la geocodificación. Si utiliza el procedimiento almacenado db2gse.gse_disable_autogc, los puede volver a activar llamando al procedimiento almacenado db2gse.gse_enable_autogc.
3. Si desea ejecutar un geocodificador en modalidad de proceso por lotes para llenar una columna espacial que tiene un índice, inhabilite o elimine antes el índice. De lo contrario, si el índice permanece operativo mientras se ejecuta el geocodificador, el rendimiento se degradará significativamente. Si utiliza el Centro de control, puede inhabilitar el índice desde la ventana Ejecutar geocodificador. DB2 Spatial Extender vuelve a habilitar el índice automáticamente cuando termina la geocodificación. Si utiliza un programa de aplicación, puede eliminar el índice con la sentencia SQL DROP. Si lo hace, asegúrese de anotar los parámetros del índice para poder volver a generarlo una vez terminada la geocodificación de proceso por lotes.
4. Cuando el geocodificador lee un registro de datos fuente, intenta hacer coincidir dicho registro con su correspondiente en los datos de referencia. La coincidencia debe tener cierto grado de precisión (denominado *precisión*) para que el geocodificador procese el registro. Por ejemplo, una precisión de 85 significa que la coincidencia entre un registro fuente y su correspondiente en los datos de referencia debe tener una precisión de al menos un 85 por ciento para que se procese el registro fuente.

   El usuario especifica la precisión. Observe que es posible que tenga que ajustarla. Por ejemplo, supongamos que la precisión es 100. Si muchos registros fuente contienen direcciones que son más recientes que los datos de referencia, será imposible obtener una precisión del 100 por ciento en la comparación entre estos registros y los datos de referencia. Como resultado, el geocodificador rechazará estos registros. En resumen, si un geocodificador genera datos espaciales que parecen insuficientes o poco precisos, puede resolver este problema cambiando la precisión y volviendo a ejecutar el geocodificador.

Ejecución del geocodificador en modalidad de proceso por lotes

Esta sección contiene una visión general de los pasos a seguir para ejecutar un geocodificador en modalidad de proceso por lotes desde el Centro de control. La visión general va seguida de detalles sobre cómo llevar a cabo cada paso.

Para saber qué autorización necesita para ejecutar un geocodificador en modalidad de proceso por lotes, consulte el tema "Autorización" en la página 102.

Visión general de los pasos a seguir para ejecutar un geocodificador en modalidad de proceso por lotes:

1. Abra la ventana Ejecutar geocodificador.
2. Indique qué geocodificador desea utilizar.
3. Inhabilite los objetos que puedan dificultar el rendimiento del geocodificador.
4. Especifique el número de registros a geocodificar antes de que DB2 emita un punto de compromiso.
5. Indique el modo en que desea ejecutar el geocodificador.
6. Indique a DB2 Spatial Extender que ejecute el geocodificador.

Pasos detallados para ejecutar un geocodificador en modalidad de proceso por lotes:

1. Abra la ventana Ejecutar geocodificador.
   a. Desde la ventana Centro de control, amplíe el árbol de objetos hasta encontrar la carpeta **Tablas** en su base de datos habilitada para operaciones espaciales.
   b. Pulse la carpeta **Tablas**. Se mostrarán las tablas en el panel de contenido de la parte derecha de la ventana.
   c. Pulse con el botón derecho del ratón la tabla que desee en el panel de contenido y pulse **Capas espaciales** en el menú emergente. Se abrirá la ventana Capas espaciales.
   d. En la ventana Capas espaciales:
      1) Seleccione la capa que está definida en la columna que desea llenar.
      2) Pulse el botón **Ejecutar geocodificador**. Se abrirá la ventana Ejecutar geocodificador.
2. Si desea utilizar el geocodificador por omisión, no modifique el recuadro **Nombre**, que muestra el nombre del geocodificador por omisión. Si no es así, utilice el recuadro para seleccionar el geocodificador que desee.
3. Inhabilite los objetos que puedan dificultar el rendimiento del geocodificador:

- Si la columna que desea llenar tiene un índice, seleccione el recuadro **Inhabilitar temporalmente los índices espaciales durante el proceso de geocodificación**.
- Si los activadores se han definido para que activen la geocodificación incremental para esta columna, seleccione el recuadro **Inhabilitar temporalmente activadores espaciales durante el proceso de geocodificación**.

  El índice y los activadores se volverán a habilitar automáticamente cuando pulse **Bien** en la ventana Ejecutar geocodificador.

4. Utilice el selector cíclico **Ámbito de confirmación** para especificar el número de registros a geocodificar antes de que DB2 emita una confirmación. Por ejemplo, si desea que DB2 confirme 100 registros geocodificados cada vez, especifique el número 100.

   Sugerencia: Si desea que DB2 emita una confirmación sólo después de que se hayan procesado todos los registros, especifique cero.
5. Utilice los campos del recuadro de grupo **Parámetros de geocodificador** para indicar el modo en que desea ejecutar el geocodificador:
   - Utilice el selector cíclico **Nivel de precisión** para especificar, en términos de porcentaje, la precisión que debe tener una coincidencia entre los registros fuente y sus correspondientes en los datos de referencia. Para obtener más información sobre la precisión, consulte el tema “Acerca de la geocodificación” en la página 41.
   - Si utiliza un geocodificador suministrado por un proveedor y desea utilizar las propiedades a las que da soporte, utilice el recuadro **Propiedades** para definir dichas propiedades.
   - Si desea geocodificar únicamente un subconjunto de filas de la tabla seleccionada, utilice el recuadro **Cláusula WHERE** para codificar una cláusula SELECT WHERE que especificará los criterios para las filas que desea. Esta cláusula puede hacer referencia a cualquiera de las columnas de la tabla.

     Escriba únicamente los criterios. Omita la palabra clave WHERE. Por ejemplo, si la tabla tiene una columna STATE y desea geocodificar únicamente las filas que contienen el valor MA en esta columna, escriba:

     ```
     STATE='MA'
     ```
6. Pulse **Bien** para ejecutar el geocodificador.

Importación y exportación de datos

Esta sección describe el proceso de importar y exportar datos y explica cómo utilizar el Centro de control para:

- Importar datos de un archivo de intercambio de datos a una tabla nueva o existente

- Importar datos de un archivo de intercambio de datos a una tabla existente
- Exportar datos de una tabla a un archivo de intercambio de datos

Acerca de la importación y la exportación

Esta sección lista las razones para importar y exportar datos espaciales. También describe los archivos de intercambio de datos que sirven como interfaces entre las fuentes de la exportación y los destinos de la importación.

Puede utilizar DB2 Spatial Extender para importar datos espaciales de archivos de intercambio de datos y para exportar datos a los mismos. Supongamos que tenemos los siguientes escenarios:

- Su GIS contiene datos espaciales que representan sus oficinas, clientes y otros temas relacionados con su empresa. Desea complementar estos datos con datos espaciales que representen el entorno cultural de su organización—ciudades, calles, puntos de interés, etc. Los datos que desea están disponibles en un proveedor de correlaciones. Puede utilizar DB2 Spatial Extender para importar estos datos de un archivo de intercambio de datos que suministra el proveedor.
- Desea migrar datos espaciales de un sistema Oracle a su GIS de DB2 Spatial Extender. Para ello utiliza un programa de utilidad de Oracle que carga los datos en un archivo de intercambio de datos. Luego utiliza DB2 Spatial Extender para importar los datos de este archivo a la base de datos que tiene habilitada para operaciones espaciales.
- Desea utilizar un examinador de GIS para mostrar presentaciones visuales de información espacial a los clientes. El examinador sólo necesita archivos con los que trabajar; no necesita estar conectado a una base de datos. Podría utilizar DB2 Spatial Extender para exportar los datos a un archivo de intercambio de datos y luego utilizar un programa de utilidad del examinador para cargar los datos en el examinador.

El Centro de control da soporte a dos tipos de archivos de intercambio de datos para DB2 Spatial Extender: archivos de forma y archivos de transferencia ESRI_SDE. Los archivos de forma se suelen utilizar para importar datos que se originan en sistemas de archivos y para exportar datos a archivos que se deben cargar en sistemas de archivos. Los archivos de transferencia de datos ESRI_SDE se suelen utilizar para importar datos que se originan en bases de datos ESRI.

Importación de datos a una tabla nueva o existente

Esta sección contiene una visión general de los pasos a seguir para importar datos de un archivo de forma o de transferencia de datos ESRI_SDE a una tabla nueva o existente. La visión general va seguida de detalles sobre cómo llevar a cabo cada paso.

Para saber qué autorización se necesita para importar datos de forma, consulte el tema “Autorización” en la página 91. Para saber qué autorización se necesita para importar datos ESRI_SDE, consulte el tema “Autorización” en la página 89.

Visión general de los pasos a seguir para importar datos a una tabla nueva o existente:

1. Abra la ventana Importar datos espaciales.
2. Especifique la vía de acceso, el nombre y el formato del archivo que contiene los datos que desea importar.
3. Especifique el número de registros a importar antes de cada confirmación.
4. Si desea importar datos espaciales a una tabla que se tiene que crear, suministre un nombre para esta tabla y un nombre para la columna a la que van destinados los datos. Si va a importar datos espaciales a una tabla existente, indique a qué columna van destinados los datos.
5. Especifique qué sistema de referencias espaciales se debe asociar a los datos.
6. Designe un archivo para que recopile los registros que no se puedan importar.
7. Indique a DB2 Spatial Extender que importe los datos y, si ha definido una tabla en esta ventana, que cree la tabla y que registre la columna a la que van destinados los datos como una capa.

Detalles de los pasos a seguir para importar datos a una tabla nueva o existente:

1. Abra la ventana Importar datos espaciales.
   a. Desde la ventana Centro de control, amplíe el árbol de objetos hasta encontrar la carpeta **Bases de datos** bajo el servidor en el que está ejecutando DB2 Spatial Extender.
   b. Pulse la carpeta **Bases de datos**. Se muestran las bases de datos en el panel de contenido en la parte derecha de la ventana.
   c. Pulse con el botón derecho del ratón la base de datos a la que desea importar datos y pulse **Spatial Extender —> Importar datos espaciales** en el menú emergente. Se abrirá la ventana Importar datos espaciales.
2. Especifique la vía de acceso, el nombre y el formato del archivo que contiene los datos que desea importar:
   a. Utilice el campo **Nombre de archivo** para especificar la vía de acceso y el nombre.
   b. Utilice el recuadro **Formato de archivo** para especificar el formato. El formato puede ser:

      Forma Éste es el valor por omisión.

ESRI_SDE

Si especifica este formato, el campo **Nombre de referencia espacial** adopta por omisión el nombre del sistema de referencias espaciales asociado a este formato.

3. Utilice el campo **Ámbito de confirmación** para especificar el número de registros que desea importar antes de cada confirmación. Por ejemplo, para solicitar a DB2 que confirme 100 registros cada vez, especifique el número 100.

   Sugerencia: Si desea que DB2 emita una confirmación sólo después de que se hayan procesado todos los registros, especifique cero.

4. Especifique la tabla y columna a las que van destinados los datos.
   a. Utilice el recuadro **Esquema de capa** para especificar el esquema correspondiente a la tabla en la que se van a importar los datos.
   b. Especifique la tabla y la columna:
      - Si la tabla aún no existe:
        1) En el campo **Tabla de capa**, escriba un nombre para la tabla.
        2) En el campo **Columna de capa**, escriba un nombre para la columna que va a contener los datos importados. DB2 Spatial Extender registrará automáticamente esta columna como una capa.
      - Si la tabla ya existe:
        1) En el campo **Tabla de capa**, especifique la tabla. Ya debe contener la columna en la que desea colocar los datos importados. Además, esta columna ya debe estar registrada como una capa.
        2) En el campo **Columna de capa**, especifique el nombre de la columna a la que van destinados los datos importados.
5. En el campo **Nombre de referencia espacial**, escriba o seleccione el sistema de referencias espaciales que se debe asociar a estos datos. (Si los datos van a proceder de un archivo de transferencia ESRI_SDE, el nombre del sistema de referencias espaciales asociado se muestra automáticamente en el campo.)
6. En el campo **Archivo de excepciones**, especifique la vía de acceso y el nombre correspondiente a un archivo nuevo en el que se recopilarán los registros que no se puedan importar. Luego puede arreglar estos registros e importarlos desde este archivo.

   DB2 Spatial Extender creará este archivo; no especifique uno que ya exista.
7. Pulse **Bien** para importar los datos. Además, si ha suministrado un nombre para una tabla que aún no existe, esta tabla se creará y la columna a la que van destinados los datos se registrará como una capa. También se creará el archivo de excepciones especificado.

Importación de datos a una tabla existente

Esta sección contiene una visión general de los pasos a seguir para importar datos de un archivo de forma o de transferencia de datos ESRI_SDE a una tabla existente. La visión general va seguida de detalles sobre cómo llevar a cabo cada paso.

Para saber qué autorización se necesita para importar datos de forma, consulte el tema "Autorización" en la página 91. Para saber qué autorización se necesita para importar datos ESRI_SDE, consulte el tema "Autorización" en la página 89.

Visión general de los pasos a seguir para importar datos a una tabla existente:

1. Abra la ventana Importar datos espaciales.
2. Especifique la vía de acceso y el nombre del archivo que contiene los datos que desea importar.
3. Especifique el número de registros a importar antes de cada confirmación.
4. Especifique la columna que debe contener los datos espaciales importados.
5. Especifique qué sistema de referencias espaciales se debe asociar a estos datos.
6. Designe un archivo para que recopile los registros que no se puedan importar.
7. Indique a DB2 Spatial Extender que importe los datos y, si ha especificado una columna que aún no se ha creado, que la cree y que la registre como una capa.

Detalles de los pasos a seguir para importar datos a una tabla existente:

1. Abra la ventana Importar datos espaciales.
   a. Desde la ventana Centro de control, amplíe el árbol de objetos hasta encontrar la carpeta **Tablas** correspondiente a la base de datos a la que desea importar datos.
   b. Pulse la carpeta **Tablas**. Se mostrarán las tablas en el panel de contenido de la parte derecha de la ventana.
   c. Pulse con el botón derecho del ratón la tabla en la que desea importar datos y pulse **Spatial Extender** —> **Importar datos espaciales** en el menú emergente. Se abrirá la ventana Importar datos espaciales.
2. En el recuadro **Nombre de archivo**, especifique la vía de acceso y el nombre del archivo que contiene los datos a importar.
3. Utilice el recuadro **Ámbito de confirmación** para especificar el número de registros que desea importar antes de cada confirmación. Por ejemplo, para solicitar a DB2 que confirme 100 registros cada vez, especifique el número 100.

Sugerencia: Si desea que DB2 emita una confirmación sólo después de que se hayan procesado todos los registros, especifique cero.

4. Especifique la columna que debe contener los datos espaciales importados.
   - Si la columna aún no existe en la tabla, utilice el recuadro **Columna de capa** para escribir un nombre para la columna.
   - Si la columna ya existe, utilice el recuadro **Columna de capa** para seleccionar o escribir el nombre de la columna.
5. Utilice el recuadro **Nombre de referencia espacial** para especificar qué sistema de referencias espaciales se debe asociar a los datos importados.
   - Si está añadiendo una columna a una tabla, escriba o seleccione el nombre del sistema de referencias espaciales.
   - Si los datos importados van destinados a una columna existente, no modifique el recuadro **Nombre de referencia espacial**. Este muestra el nombre del sistema de referencias espaciales por omisión.
6. En el campo **Archivo de excepciones**, especifique la vía de acceso y el nombre correspondiente a un archivo nuevo en el que se recopilarán los registros que no se puedan importar. Luego puede arreglar estos registros e importarlos desde este archivo.

   DB2 Spatial Extender creará este archivo; no especifique uno que ya exista.
7. Pulse **Bien** para importar los datos. Además, si ha especificado una columna que aún no existe, esta columna se creará y se registrará como una capa. También se creará el archivo de excepciones especificado.

Exportación de datos a un archivo de forma

Esta sección contiene una visión general de los pasos a seguir para exportar datos a un archivo de forma. La visión general va seguida de detalles sobre cómo llevar a cabo cada paso.

Para saber qué autorización se necesita para llevar a cabo estos pasos, consulte el tema “Autorización” en la página 87.

Visión general de los pasos a seguir para exportar datos de un archivo de forma:

1. Abra la ventana Exportar datos espaciales.
2. Especifique la columna que contiene los datos espaciales que se deben exportar.
3. Si desea exportar un subconjunto de filas de datos, identifique este subconjunto ante DB2 Spatial Extender.
4. Especifique la vía de acceso y el nombre del archivo al que va a exportar los datos.
5. Indique a DB2 Spatial Extender que exporte los datos.

Detalles de los pasos a seguir para exportar datos a un archivo de forma:

1. Abra la ventana Exportar datos espaciales.
   a. En la ventana Centro de control, amplíe el árbol de objetos hasta encontrar la carpeta **Tablas** o **Vistas** de la base de datos que contiene datos espaciales:
   b. Pulse la carpeta **Tablas** o **Vistas**. Se mostrarán las tablas o vistas en el panel de contenido de la parte derecha de la ventana.
   c. Pulse con el botón derecho del ratón la tabla o vista que contiene los datos a exportar y pulse **Spatial Extender** —> **Exportar datos espaciales** en el menú emergente. Se abrirá la ventana Exportar datos espaciales.
2. En el campo **Columna de capa**, especifique el nombre de la columna que contiene los datos espaciales a exportar.
3. Si desea exportar un subconjunto de filas de la tabla, utilice el recuadro **Cláusula WHERE** para escribir una cláusula WHERE que especifique los criterios correspondientes a las filas que desea. En esta cláusula sólo puede hacer referencia a columnas de la tabla o vista de la que está exportando datos.

   Escriba únicamente los criterios. Omita la palabra clave WHERE. Por ejemplo, si la tabla o vista tiene una columna STATE y desea geocodificar únicamente las filas que contienen el valor MA en esta columna, escriba:

   ```
   STATE='MA'
   ```
4. En el campo **Nombre de archivo**, especifique la vía de acceso y el nombre del archivo al que está exportando datos.
5. Pulse **Bien** para exportar los datos.

Capítulo 6. Creación de índices espaciales

Este capítulo explica cómo utilizar el Centro de control para crear un índice para datos espaciales.

Después de llenar con sus datos las columnas espaciales, ya está listo para crear un índice espacial. Las típicas estructuras de índice, como un árbol B, realizan clasificaciones lineales, de una dimensión, de los datos de una tabla. Los datos de una tabla habilitados para operaciones espaciales no se almacenan como una sola entrada, sino que tienen dos dimensiones. Por ejemplo, las geometrías espaciales, como un polígono, constan de varios valores de coordenadas en una columna o capa espacial. Puesto que un índice tipo árbol B no puede manejar tipos de datos espaciales, DB2 Spatial Extender ha creado su propia tecnología de indexación denominada *índice de cuadrícula*. El índice de cuadrícula se basa en el índice tipo árbol B, pero mejorado para manejar datos de dos dimensiones y realizar indexación de columnas espaciales. El índice de cuadrícula da soporte a tres capas y está diseñado para ofrecer un buen rendimiento en una amplia gama de objetos, tamaños y distribuciones de datos. Para obtener más información sobre índices espaciales, consulte el “Capítulo 12. Índices espaciales” en la página 121.

Para saber qué autorización se necesita para crear un índice espacial, consulte el tema “Autorización” en la página 81.

Utilización del Centro de control para crear un índice espacial

Para crear un índice espacial mediante el Centro de control:

1. En el árbol de objetos, seleccione la carpeta **Tablas**. Todas las tablas existentes se mostrarán en el panel de contenido.
2. En el panel de contenido, pulse con el botón derecho del ratón la tabla para la que desea crear un índice y pulse **Spatial Extender** —> **Índices espaciales** en el menú emergente. Se abrirá la ventana Índices espaciales.
3. En la ventana Índices espaciales, pulse **Crear**. Se abrirá la ventana Crear índice espacial.
4. En el campo **Nombre**, escriba el nombre del nuevo índice espacial que desea crear.

   Nota: No hace falta que especifique un esquema. DB2 Spatial Extender añadirá automáticamente el esquema y creará un nombre calificado al completo.

5. En el campo **Columna de capa**, seleccione la capa para la que está creando un índice.

   Una capa es una columna espacial definida o registrada ante DB2 Spatial Extender.

6. En los campos **Tamaño de cuadrícula**, escriba el valor de tamaño de cuadrícula que desea asignar a cada campo.

   Los niveles de cuadrícula, **Fino**, **Medio** y **Grueso**, se entran aumentando el tamaño de celda. Por lo tanto, el segundo nivel debe tener un tamaño de celda superior al del primero, y el tercero uno superior al del segundo.

Determinación de los tamaños de celda de cuadrícula

La determinación del tamaño de cuadrícula correcto se realiza mediante un proceso de prueba y error. Se recomienda definir un tamaño de celda en relación al tamaño aproximado del objeto que está indexando. Los tamaños de celda insuficientes o excesivos pueden reducir el rendimiento. Los tamaños insuficientes afectan a la relación clave/objeto durante una búsqueda de índice. En este caso, se crean demasiadas claves y se devuelve un alto número de candidatos. Para tamaños de cuadrícula excesivos, la búsqueda de índice inicial devuelve un pequeño número de candidatos, pero es posible que se reduzca el rendimiento durante la exploración final de la tabla.

Para obtener más información sobre cómo seleccionar tamaños de celda de cuadrícula y el número de niveles de cuadrícula, consulte el tema “Selección del tamaño de celda de cuadrícula” en la página 128.

Capítulo 7. Recuperación y análisis de información espacial

Después de construir índices espaciales, ya se pueden utilizar las tablas espaciales. Este capítulo trata temas relacionados con la recuperación y análisis de datos espaciales. Contiene una visión general de diversos métodos de recuperación y ofrece ejemplos de consultas de tablas que utilizan funciones espaciales.

Métodos de realizar análisis espaciales

Puede realizar análisis espaciales utilizando SQL y funciones espaciales con cualquiera de los siguientes entornos de programación:

- Un geoexaminador (por ejemplo, ArcExplorer de ESRI).

  Para obtener más información sobre la utilización de ArcExplorer, consulte el manual *Using ArcExplorer*, disponible en el sitio Web de ESRI en la dirección http://www.esri.com.
- Sentencias de SQL interactivas.
- Aplicaciones desarrolladas por el usuario (por ejemplo, ODBC, JDBC y SQL intercalado).

Las aplicaciones se pueden ejecutar desde el Centro de mandatos de DB2, la ventana Mandato de DB2 o el procesador de línea de mandatos.

Creación de una consulta espacial

Esta sección explica cómo crear consultas espaciales que utilizan predicados y funciones espaciales.

Funciones espaciales y SQL

DB2 Spatial Extender incluye funciones que realizan varias operaciones sobre datos espaciales. Los ejemplos de esta sección le muestran cómo utilizar funciones espaciales para crear sus propias consultas espaciales.

La Tabla 3 contiene una lista de funciones espaciales y los tipos de operaciones que pueden realizar.

Tabla 3. Funciones y operaciones espaciales

Tipo de función	Ejemplo de operación
Cálculo	Calcular la distancia entre dos puntos
Comparación	Buscar todos los clientes ubicados en una zona de inundaciones
Intercambio de datos	Convertir datos en formatos soportados
Transformación	Añadir un radio de cinco millas a un punto

Para obtener más información sobre funciones espaciales, consulte el "Capítulo 13. Geometrías y funciones espaciales asociadas" en la página 131 y el "Capítulo 14. Funciones espaciales para consultas SQL" en la página 169.

Ejemplo 1: Comparación

La siguiente consulta busca la distancia media de los clientes de cada centro comercial. Las funciones espaciales utilizadas en este ejemplo son ST_Distance y ST_Within.

```
SELECT s.id, AVG(db2gse.ST_Distance(c.location,s.location))
FROM customers c, stores s
WHERE db2gse.ST_Within(c.location,s.zone)=1
GROUP BY s.id
```

Ejemplo 2: Intercambio de datos

La siguiente consulta busca las ubicaciones de los clientes que viven en el área de la bahía de San Francisco. Las funciones espaciales utilizadas en este ejemplo son ST_AsText (intercambio de datos) y ST_Within. ST_AsText convierte los datos espaciales de la columna c.location en el formato OGC TEXT.

```
SELECT db2gse.ST_AsText(c.location,cordref(1))
FROM customers c
WHERE db2gse.ST_Within(c.location,:BayArea)=1
```

Ejemplo 3: Cálculo

La siguiente consulta busca todas las calles de más de 10,5 millas. La función espacial utilizada es ST_Length.

```
SELECT s.name,s.id
FROM street s
WHERE db2gse.ST_Length(s.path) > 10.5
```

Ejemplo 4: Transformación

Esta consulta busca los clientes que viven dentro de un área con riesgo de inundación o a menos de 2 millas del límite del área con peligro de inundación. Las funciones espaciales utilizadas en este ejemplo son ST_Buffer (transformación) y ST_Within.

```
SELECT c.name,c.phoneNo,c.address
FROM customers c
WHERE db2gse.ST_Within(c.location,ST_Buffer(:floodzone,2))=1
```

Predicados espaciales y SQL

Un grupo especializado de funciones espaciales denominadas predicados espaciales permiten mejorar el rendimiento de las consultas. Los predicados espaciales, como ST_Overlaps, que compara dos polígonos para ver si se solapan, pueden resultar caros de ejecutar por requisitos tanto de tiempo como de memoria. Por lo tanto, las técnicas de optimización para minimizar el coste de ejecución resultan muy importantes. El optimizador de consultas de DB2 utiliza el índice espacial para mejorar el rendimiento de las consultas cuando el usuario utiliza predicados espaciales según las reglas que se describen más adelante en esta sección. Para obtener más información sobre predicados espaciales, consulte el tema “Funciones de predicado” en la página 145. Los predicados espaciales utilizados para aprovechar el índice espacial son:

- ST_Contains
- ST_Crosses
- ST_Disjoint
- ST_Distance
- ST_Envelope
- ST_Equals
- ST_Intersects
- ST_Overlaps
- ST_Touches
- ST_Within

Para ver una lista completa de todas las funciones y predicados espaciales, consulte el “Capítulo 14. Funciones espaciales para consultas SQL” en la página 169.

Reglas para el aprovechamiento del índice

Las siguientes reglas sirven para optimizar las consultas espaciales que utilizan predicados espaciales:

- El predicado se debe utilizar en la cláusula WHERE.
- El predicado debe quedar a la izquierda de la comparación. Por ejemplo:
  ```
  WHERE db2gse.ST_Within(c.location,:BayArea)=1
  ```
- Las comparaciones de igualdad deben utilizar la constante de entero 1.
  ```
  WHERE db2gse.ST_Within(c.location,:BayArea)=1
  ```
- Debe haber una columna espacial que se utilice en el predicado como el destino de la búsqueda y debe haber un índice espacial creado en dicha columna.

Ejemplos de aprovechamiento del índice

La Tabla 4 muestra formas correctas e incorrectas de crear consultas espaciales para aprovechar el índice espacial.

Tabla 4. Reglas para el aprovechamiento del índice

Consulta espacial	Regla violada

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Within(c.location,:BayArea)=1`
	En este ejemplo no se viola ninguna condición.

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Distance(c.location,:SanJose)<10`
	En este ejemplo no se viola ninguna condición.

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Length(c.location)>10`
	El predicado se debe utilizar en la cláusula WHERE. (ST_Length es una función espacial, pero *no* un predicado.)

`SELECT *`
`FROM customers c`
`WHERE 1=db2gse.ST_Within(c.location,:BayArea)`
	El predicado debe quedar a la izquierda de la comparación.

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Within(c.location,:BayArea)=2`
	Las comparaciones de igualdad deben utilizar la constante de entero 1.

`SELECT *`
`FROM customers c`
`WHERE db2gse.ST_Within(:SanJose,:BayArea)=1`
	Debe haber una columna espacial que se utilice en el predicado como el destino de la búsqueda y debe haber un índice espacial creado en dicha columna. (SanJose y BayArea no son columnas espaciales y, por lo tanto, no pueden tener asociado un índice espacial.)

Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender

Este capítulo explica cómo utilizar el programa de ejemplo de DB2 Spatial Extender para escribir aplicaciones para trabajar con información espacial y personalizarla. Se incluyen los siguientes temas:

- Utilización del programa de ejemplo
- Los pasos del programa de ejemplo

Utilización del programa de ejemplo

El programa de ejemplo de DB2 Spatial Extender facilita la programación de aplicaciones. Con el programa de ejemplo, puede:

- Automatizar los procedimientos espaciales rutinarios
- Cortar y pegar código de ejemplo en sus propias aplicaciones
- Entender los pasos que suelen ser necesarios para crear y mantener una base de datos habilitada para operaciones espaciales

Utilice el programa de ejemplo para codificar tareas complejas para DB2 Spatial Extender, por ejemplo para escribir una aplicación que utilice la interfaz de base de datos para llamar a procedimientos almacenados de DB2 Spatial Extender. Desde el programa de ejemplo, puede copiar y personalizar sus aplicaciones. Si no está familiarizado con los pasos de programación a seguir para DB2 Spatial Extender, puede ejecutar el programa de ejemplo, que le enseña cada paso con detalle. Antes debe crear el programa de ejemplo. Para ello utilice el archivo makefile de ejemplo. Para ver instrucciones sobre cómo crear y ejecutar el programa de ejemplo, consulte “Verificación de la instalación” en la página 20.

Los pasos del programa de ejemplo

La Tabla 5 en la página 60 muestra los pasos del programa de ejemplo, los procedimientos almacenados asociados y una descripción de cada paso. Las funciones C para invocar los procedimientos almacenados se muestran en la columna Acción de la Tabla 5 en la página 60 y aparecen entre paréntesis. Para obtener más información sobre los procedimientos almacenados, consulte el “Capítulo 9. Procedimientos almacenados” en la página 69. El programa de ejemplo se basa en los escenarios descritos en el tema “Escenario: una compañía de seguros actualiza su GIS” en la página 12.

Tabla 5. Programa de ejemplo de DB2 Spatial Extender

Pasos del programa de ejemplo	Acción	Descripción
Habilitar / inhabilitar la base de datos espacial	1. Habilitar la base de datos espacial (gseEnableDB) 2. Inhabilitar la base de datos espacial (gseDisableDB) 3. Habilitar la base de datos espacial (gseEnableDB)	1. Este es el primer paso a seguir para poder utilizar DB2 Spatial Extender. Una base de datos habilitada para operaciones espaciales tiene una serie de tipos espaciales, una serie de funciones espaciales, una serie de predicados espaciales, un nuevo tipo de índice y una serie de vistas y tablas de administración. 2. Este paso se suele llevar a cabo cuando ha habilitado funciones espaciales para la base de datos equivocada. Cuando inhabilita una base de datos espacial, elimina una serie de tipos espaciales, una serie de funciones espaciales, una serie de predicados espaciales, un nuevo tipo de índice y una serie de vistas y tablas de administración. **Nota:** La inhabilitación de la base de datos fallará si hay objetos creados que dependen de los objetos creados por el procedimiento de habilitación de la base de datos. Por ejemplo, la creación de una tabla con una columna espacial del tipo ST_Point hará que falle la inhabilitación de la base de datos. Esto sucede porque la tabla depende del tipo ST_Point, el cual debe eliminar el procedimiento de inhabilitación de la base de datos. 3. Igual que 1.

Tabla 5. Programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos del programa de ejemplo	Acción	Descripción
Registrar sistemas de referencias espaciales	1. Registrar el sistema de referencias espaciales para la columna LOCATION de la tabla CUSTOMERS (gseEnableSref) 2. Registrar el sistema de referencias espaciales para la columna LOCATION de la tabla OFFICES (gseEnableSref) 3. Desregistrar el sistema de referencias espaciales para la columna LOCATION de la tabla OFFICES (gseDisableSref) 4. Volver a registrar el sistema de referencias espaciales para las columnas ZONE de la tabla OFFICES (gseEnableSref)	1. Este paso define un nuevo sistema de referencias espaciales (SRS) cuya finalidad es interpretar los datos espaciales de la tabla CUSTOMERS. Un sistema de referencias espaciales incluye datos de geometría en un formato que se puede almacenar en una columna de una base de datos habilitada para operaciones espaciales. Después de que un SRS se registra ante una determinada capa, las coordenadas aplicables a dicha capa se pueden almacenar en la columna de la tabla CUSTOMERS asociada. 2. Este paso define un nuevo sistema de referencias espaciales (SRS) cuya finalidad es interpretar los datos espaciales de la capa OFFICES. Cada capa de tabla debe tener un SRS definido. Es posible que las capas de la tabla OFFICES necesiten un SRS asociado distinto del de la capa de la tabla CUSTOMERS. 3. Este paso sólo se lleva a cabo si el usuario especifica los parámetros de SRS incorrectos ante la capa o columna espacial. Cuando el usuario desregistra un SRS correspondiente a la capa de la tabla OFFICES, elimina la definición con sus parámetros asociados. 4. Este paso define un nuevo sistema de referencias espaciales (SRS) cuya finalidad es interpretar los datos espaciales de la capa OFFICES.

Tabla 5. Programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos del programa de ejemplo	Acción	Descripción
Crear las tablas espaciales	1. Modificar la tabla CUSTOMERS, añadiendo la columna LOCATION (gseSetupTables) 2. Crear la tabla OFFICES (gseSetupTables)	1. La tabla CUSTOMERS representa datos comerciales que se han almacenado en la base de datos durante varios años. La sentencia ALTER TABLE añade una nueva columna (LOCATION) del tipo ST_Point. Esta columna se llenará geocodificando las columnas de dirección en un paso posterior. 2. La tabla OFFICES representa, entre otros datos, la zona de ventas correspondiente a cada oficina de una compañía de seguros. La tabla entera se llenará con los datos de atributo procedentes de una base de datos que no es de DB2 en un paso posterior. Este paso incluye la importación de datos de atributo en la tabla OFFICES procedentes del archivo SHAPE.
Registrar capas espaciales	1. Registrar la columna LOCATION en la tabla CUSTOMERS como una capa (gseRegisterLayer) 2. Registrar la columna ZONE de la tabla OFFICES como una capa (gseRegisterLayer)	Estos pasos registran las columnas LOCATION y ZONE como capas ante DB2 Spatial Extender. Para que los programas de utilidad de DB2 Spatial Extender (por ejemplo, el geocodificador) puedan llenar o acceder a una columna espacial, tiene que registrarla como una capa.

Tabla 5. Programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos del programa de ejemplo	Acción	Descripción
Llenar las capas espaciales	1. Geocodificar los datos de dirección correspondientes a la columna LOCATION de la tabla CUSTOMERS (gseRunGC) 2. Cargar la tabla OFFICES utilizando la modalidad de agregación (gseImportShape) 3. Cargar la tabla HAZARD_ZONE utilizando la modalidad de creación (gseImportShape)	1. Este paso lleva a cabo una geocodificación de proceso por lotes invocando el programa de utilidad del geocodificador. La geocodificación de proceso por lotes se suele realizar cuando una parte significativa de la tabla se tiene que geocodificar o se tiene que volver a geocodificar. 2. Este paso carga la tabla OFFICES con datos espaciales existentes en el formato de un archivo SHAPE. Puesto que la tabla OFFICES existe y la capa OFFICES/ZONE está registrada, el programa de utilidad de carga agregará los nuevos registros a una tabla existente. 3. Este paso carga la capa HAZARD_ZONE con datos espaciales existentes en el formato de un archivo SHAPE. Puesto que ni la tabla ni la capa existen, el programa de utilidad creará la tabla y registrará la capa antes de que se carguen los datos.
Habilitar índices espaciales	1. Habilitar el índice espacial para la columna LOCATION de la tabla (gseEnableIdx) 2. Habilitar el índice espacial para la columna ZONE de la tabla OFFICES (gseEnableIdx) 3. Habilitar el índice espacial para la columna LOCATION de la tabla OFFICES (gseEnableIdx) 4. Habilitar el índice espacial para la columna BOUNDRY de la tabla HAZARD_ZONE (gseEnableIdx)	Estos pasos habilitan el índice espacial para las tablas CUSTOMERS, OFFICES y HAZARD_ZONE.
Habilitar la geocodificación automática	1. Habilitar la geocodificación automática para las columnas LOCATION y ADDRESS de la tabla CUSTOMERS (gseEnableAutoGC)	Esta paso activa la invocación automática del geocodificador. La geocodificación automática permite que las columnas LOCATION y ADDRESS de la tabla CUSTOMERS estén sincronizadas entre sí para futuras operaciones de inserción y actualización.

Tabla 5. Programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos del programa de ejemplo	Acción	Descripción
Insertar / actualizar la tabla CUSTOMERS	1. Insertar algunos registros con otra calle (gseInsDelUpd) 2. Actualiza algunos registros con una dirección nueva (gseInsDelUpd)	Estos pasos demuestran una inserción y actualización en la columna LOCATION de la tabla CUSTOMERS. Una vez habilitada la geocodificación automática, la información procedente de la columna ADDRESS se geocodifica automáticamente cuando se inserta o actualiza en la columna LOCATION. Este proceso se ha habilitado en el paso anterior.
Inhabilitar la geocodificación automática	1. Inhabilitar la geocodificación automática para la capa CUSTOMERS (gseDisableAutoGC) 2. Inhabilitar el índice espacial para la capa CUSTOMERS (gseDisableIdxCustomersLayer)	Estos pasos inhabilitan la invocación automática del geocodificador y del índice espacial como preparación para el siguiente paso (el siguiente paso incluye una nueva geocodificación de la tabla CUSTOMERS entera). Si está cargando gran cantidad de geodatos, se recomienda inhabilitar el índice espacial antes de cargar los datos y luego habilitarlo una vez finalizada la carga.
Volver a geocodificar la tabla CUSTOMERS	1. Geocodificar de nuevo la capa CUSTOMERS con un nivel de precisión más bajo – 90% en lugar de 100% (gseRunGC) 2. Volver a habilitar el índice espacial para la capa CUSTOMERS (gseEnableIdx) 3. Volver a habilitar la geocodificación automática con un nivel de precisión más bajo – 90% en lugar de 100% (gseEnableAutoGC)	Estos pasos ejecutan de nuevo el geocodificador en modalidad de proceso por lotes, vuelven a habilitar la geocodificación automática con un nuevo nivel de precisión y vuelven a habilitar el índice espacial y la geocodificación automática. Esta acción se recomienda cuando el administrador espacial advierte un alto nivel de errores en el proceso de geocodificación. Si el nivel de precisión se define en 100%, es posible que no pueda geocodificar una dirección porque no encuentra una dirección correspondiente en los datos de referencia. Al reducir el nivel de precisión, el geocodificador tiene más probabilidades de encontrar datos coincidentes. Una vez se ha vuelto a geocodificar la tabla en modalidad de proceso por lotes, tanto la geocodificación automática como el índice espacial se vuelven a habilitar para facilitar el mantenimiento incremental del índice espacial y de la columna espacial para futuras operaciones de inserción y actualización.

Tabla 5. Programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos del programa de ejemplo	Acción	Descripción
Crear una vista y registrar sus columnas espaciales como capas de vista	1. Crear una vista, HIGHRISK_CUSTOMERS, basada en la unión de la tabla CUSTOMERS y la tabla HAZARD_ZONE (gseCreateView) 2. Registrar las columnas espaciales de la vista como capas (gseRegisterLayer)	Estos pasos crean una vista y registran sus columnas espaciales como capas de vista.
Realizar análisis espacial	1. Buscar la distancia media de los clientes de cada oficina (ST_Within, ST_Distance) 2. Buscar el ingreso y la prima media por cliente para cada oficina (ST_Within) 3. Buscar clientes que no están cubiertos por ninguna oficina existente (ST_Within) 4. Buscar el número de zonas peligrosas que solapa cada zona de oficinas (ST_Overlaps) 5. Buscar la oficina más cercana a la ubicación de un determinado cliente, suponiendo que la oficina se encuentra en el centro de la zona de oficinas (ST_Distance, ST_Centroid) 6. Buscar clientes cuya ubicación es cercana al límite de una determinada zona peligrosa (ST_Buffer, ST_Overlaps) 7. Buscar aquellos clientes de alto riesgo que están cubiertos por una determinada oficina (Todos los pasos utilizan gseRunSpatialQueries)	Estos pasos realizan un análisis espacial utilizando las funciones y predicados espaciales en lenguaje SQL de DB2. El optimizador de consultas de DB2 aprovecha el índice espacial de las columnas espaciales para mejorar el rendimiento de las consultas siempre que es posible.

Tabla 5. Programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos del programa de ejemplo	Acción	Descripción
Exportar capas espaciales a archivos	Exportar la capa highRiskCustomers (gseExportShape)	El paso muestra un ejemplo de exportación de los resultados de su consulta a un archivo SHAPE. Al exportar los resultados de la consulta a otro formato de archivo se permite que la información sea utilizada por una herramienta de otro proveedor (por ejemplo, ESRI ArcInfo).

Parte 2. Material de consulta

Capítulo 9. Procedimientos almacenados

Este capítulo documenta los procedimientos almacenados que le permiten crear un sistema de información geográfica con DB2 Spatial Extender. Cuando habilita y utiliza DB2 Spatial Extender desde el Centro de control, invoca estos procedimientos almacenados de forma implícita. Por ejemplo, cuando pulsa **Bien** desde una ventana de DB2 Spatial Extender, DB2 llama al procedimiento o procedimientos almacenados asociados a dicha ventana. También puede invocar los procedimientos almacenados de forma explícita en un programa de aplicación. Se recomienda incluir el archivo de cabecera, db2gse.h, en un programa de este tipo. Este archivo contiene las definiciones de macros correspondientes a las constantes que asigna a los parámetros de los procedimientos almacenados. En AIX, se almacena en el directorio $DB2INSTANCE/sqllib/include/. En Windows NT, se almacena en el directorio %DB2PATH%\include\.

Atención:
Todas las constantes de series de caracteres correspondientes a parámetros de entrada de procedimientos almacenados son sensibles a mayúsculas y minúsculas. Para ver qué parámetros necesitan estas constantes, consulte las tablas de este capítulo.

Para poder invocar un procedimiento almacenado, de forma implícita o explícita, debe estar conectado a la base de datos en la que está instalado DB2 Spatial Extender. El primer procedimiento almacenado que va a utilizar es db2gse.gse_enable_db. Habilita la base de datos para operaciones espaciales. Sólo puede utilizar los otros procedimientos almacenados cuando haya habilitado la base de datos.

Las implantaciones de procedimientos almacenados se archivan en la biblioteca db2gse del servidor de DB2 Spatial Extender.

Puede utilizar la siguiente lista para buscar los procedimientos almacenados por sus nombres o por las tareas que llevan a cabo. La primera lista presenta los nombres:

- “db2gse.gse_enable_idx” en la página 81
- “db2gse.gse_enable_sref” en la página 84
- “db2gse.gse_export_shape” en la página 87
- “db2gse.gse_import_sde” en la página 89
- “db2gse.gse_import_shape” en la página 91
- “db2gse.gse_register_gc” en la página 93
- “db2gse.gse_register_layer” en la página 95
- “db2gse.gse_run_gc” en la página 102
- “db2gse.gse_unregist_gc” en la página 104
- “db2gse.gse_unregist_layer” en la página 105

La siguiente lista presenta las tareas que llevan a cabo los procedimientos almacenados.

- Creación de un índice para una columna espacial (consulte “db2gse.gse_enable_idx” en la página 81).
- Creación de un sistema de referencias espaciales (consulte “db2gse.gse_enable_sref” en la página 84).
- Inhabilitación de un geocodificador para que no mantenga automáticamente las columnas espaciales sincronizadas con sus correspondientes columnas de atributo (consulte “db2gse.gse_disable_autogc” en la página 72).
- Inhabilitación del soporte de operaciones espaciales en una base de datos (consulte “db2gse.gse_disable_db” en la página 74).
- Eliminación de un sistema de referencias espaciales (consulte “db2gse.gse_disable_sref” en la página 75).
- Habilitación de una base de datos para que dé soporte a operaciones espaciales (consulte “db2gse.gse_enable_db” en la página 80).
- Habilitación de un geocodificador para que mantenga automáticamente las columnas espaciales sincronizadas con sus correspondientes columnas de atributo (consulte “db2gse.gse_enable_autogc” en la página 76).
- Exportación de una capa y de su tabla asociada a un archivo de forma (consulte “db2gse.gse_export_shape” en la página 87).
- Importación de una capa y de su tabla asociada de un archivo de transferencia ESRI_SDE (consulte “db2gse.gse_import_sde” en la página 89).
- Importación de una capa y de su tabla asociada de un archivo de forma (consulte “db2gse.gse_import_shape” en la página 91).
- Registro de un codificador que no es el codificador por omisión (consulte “db2gse.gse_register_gc” en la página 93).
- Registro de una columna espacial como una capa (consulte “db2gse.gse_register_layer” en la página 95).
- Ejecución de un geocodificador en modalidad de proceso por lotes (consulte “db2gse.gse_unregist_gc” en la página 104).

- Desregistro de un codificador que no es el codificador por omisión (consulte “db2gse.gse_unregist_layer” en la página 105).
- Desregistro de una capa (consulte “db2gse.gse_unregist_layer” en la página 105).

Para obtener información sobre las secuencias en las que puede llevar a cabo estas tareas, consulte el “Capítulo 1. Acerca de DB2 Spatial Extender” en la página 3 y el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59.

db2gse.gse_disable_autogc

Utilice este procedimiento almacenado para eliminar o inhabilitar temporalmente los activadores que mantienen una columna espacial sincronizada con su columna o columnas de atributo. Por ejemplo, se recomienda inhabilitar los activadores mientras geocodifica los valores en la columna o columnas de atributo en modalidad de proceso por lotes. Para obtener más información sobre este tema, consulte “Acerca de la geocodificación” en la página 41.

Para ver un ejemplo del código que sirve para invocar este procedimiento almacenado, consulte la función C gseDisableAutoGc en el programa de ejemplo. Para obtener información sobre este programa, consulte el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59.

Autorización

El ID de usuario bajo el cual se invoca este procedimiento almacenado debe tener autorización en forma de autoridad, privilegio o grupo de privilegios; en concreto:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla en la que están definidos los activadores que se están eliminando o inhabilitando temporalmente.
- El privilegio CONTROL sobre esta tabla.
- Los privilegios ALTER, SELECT y UPDATE sobre esta tabla.

Parámetros de entrada

Tabla 6. Parámetros de entrada para el procedimiento almacenado db2gse.gse_disable_autogc.

Nombre	Tipo de datos	Descripción
operMode	SMALLINT	Indica si los activadores se tienen que eliminar o inhabilitar temporalmente. Este parámetro no se puede anular. **Comentario**: Para eliminar activadores, utilice la macro GSE_AUTOGC_DROP. Para inhabilitarlos temporalmente, utilice la macro GSE_AUTOGC_INVALIDATE. Para saber cuáles son los valores asociados a estas macros, consulte el archivo db2gse.h. En AIX, este archivo se almacena en el directorio $DB2INSTANCE/sqllib/include/. En Windows NT, se almacena en el directorio %DB2PATH%\include\.

Tabla 6. Parámetros de entrada para el procedimiento almacenado db2gse.gse_disable_autogc. (continuación)

Nombre	Tipo de datos	Descripción
layerSchema	VARCHAR (30)	Nombre del esquema al que pertenece la tabla o vista especificada en el parámetro layerTable. Este parámetro no se puede anular. **Comentario:** Si no especifica un valor para el parámetro layerSchema, este adoptará como valor por omisión el ID de usuario bajo el que se invoca el procedimiento almacenado db2gse.gse_disable_autogc.
layerTable	VARCHAR(128)	Nombre de la tabla en la que están definidos los activadores que desea eliminar o inhabilitar temporalmente. Este parámetro no se puede anular.
layerColumn	VARCHAR(128)	Nombre de la columna habilitada para operaciones espaciales que se mantiene mediante los activadores que desea eliminar o inhabilitar temporalmente. Este parámetro no se puede anular.

Parámetros de salida

Tabla 7. Parámetros de salida para el procedimiento almacenado db2gse.gse_disable_autogc.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

db2gse.gse_disable_db

Utilice este procedimiento almacenado para eliminar recursos que permiten a DB2 Spatial Extender almacenar datos espaciales y dar soporte a operaciones realizadas sobre estos datos.

El objetivo de este procedimiento almacenado consiste en ayudarle a resolver problemas o dudas que se presentan tras habilitar la base de datos para operaciones espaciales pero *antes* de añadir a la misma datos o columnas de la tabla espacial. Por ejemplo, si, tras habilitar una base de datos para operaciones espaciales, se decide utilizar DB2 Spatial Extender para otra base de datos. Puesto que aún no ha definido ninguna columna espacial ni importado datos espaciales, puede invocar este procedimiento almacenado para eliminar todos los recursos espaciales de la primera base de datos.

Para ver un ejemplo del código que sirve para invocar este procedimiento almacenado, consulte la función C gseDisableDB en el programa de ejemplo. Para obtener información sobre este programa, consulte el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59.

Autorización

El ID de usuario bajo el que se invoca este procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos de la que se deben eliminar recursos de DB2 Spatial Extender.

Parámetros de salida

Tabla 8. Parámetros de salida para el procedimiento almacenado db2gse.gse_disable_db.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

db2gse.gse_disable_sref

Utilice este procedimiento almacenado para eliminar un sistema de referencias espaciales. Cuando se procesa este procedimiento almacenado, se elimina información sobre el sistema de referencias espaciales de la vista de catálogo DB2GSE.SPATIAL_REF_SYS. Para obtener información sobre esta vista, consulte el tema “DB2GSE.SPATIAL_REF_SYS” en la página 120.

Para ver un ejemplo del código que sirve para invocar este procedimiento almacenado, consulte la función C gseDisableSref en el programa de ejemplo. Para obtener información sobre este programa, consulte el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59.

Autorización

No se necesita.

Parámetro de entrada

Tabla 9. Parámetros de entrada para el procedimiento almacenado db2gse.gse_disable_sref.

Nombre	Tipo de datos	Descripción
srId	INTEGER	Identificador numérico del sistema de referencias espaciales que se tiene que eliminar. Este parámetro no se puede anular.

Parámetros de salida

Tabla 10. Parámetros de salida para el procedimiento almacenado db2gse.gse_disable_sref.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

Restricción

Para poder eliminar un sistema de referencias espaciales, debe desregistrar las capas que lo utilizan. Si estas capas se mantienen sin registrar, la petición de eliminar el sistema de referencias espaciales se rechazará.

db2gse.gse_enable_autogc

Utilice este procedimiento almacenado para:

- Crear activadores que mantendrán una columna espacial sincronizada con su columna o columnas de atributo asociadas. Cada vez que se insertan valores en la columna o columnas de atributo o se actualizan valores de las mismas, un activador llamará a un geocodificador registrado para que geocodifique los valores insertados o actualizados y coloque los datos resultantes en la columna espacial.
- Volver a activar los activadores después de que se hayan inhabilitado temporalmente.
- Establecer qué función se utilizará para geocodificar los valores insertados o actualizados.

Para ver un ejemplo del código que sirve para invocar este procedimiento almacenado, consulte la función C gseEnableAutoGC en el programa de ejemplo. Para obtener información sobre este programa, consulte el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59.

Autorización

El ID de usuario bajo el cual se invoca este procedimiento almacenado debe tener autorización en forma de autoridad, privilegio o grupo de privilegios; en concreto:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla en la que están definidos los activadores creados por este procedimiento almacenado.
- El privilegio CONTROL sobre esta tabla.
- Los privilegios ALTER, SELECT y UPDATE sobre esta tabla.

Parámetros de entrada

Tabla 11. Parámetros de entrada para el procedimiento almacenado db2gse.gse_enable_autogc.

Nombre	Tipo de datos	Descripción
operMode	SMALLINT	Valor que indica si los activadores que inician la geocodificación se deben crear por primera vez o se deben volver a activar tras haber sido temporalmente inhabilitados. Este parámetro no se puede anular. **Comentario:** Para crear activadores, utilice la macro GSE_AUTOGC_CREATE. Para volverlos a activar, utilice la macro GSE_AUTOGC_RECREATE. Para saber cuáles son los valores asociados a estas macros, consulte el archivo db2gse.h. En AIX, este archivo se almacena en el directorio $DB2INSTANCE/sqllib/include/. En Windows NT, se almacena en el directorio %DB2PATH%\include\.
layerSchema	VARCHAR (30)	Nombre del esquema al que pertenece la tabla especificada en el parámetro layerTable. Este parámetro no se puede anular. **Comentario:** Si no especifica un valor para el parámetro layerSchema, este adoptará como valor por omisión el ID de usuario bajo el que se invoca el procedimiento almacenado db2gse.gse_enable_autogc.
layerTable	VARCHAR(128)	Nombre de la tabla sobre la que deben operar los activadores creados o vueltos a activar por este procedimiento almacenado. Este parámetro no se puede anular.
layerColumn	VARCHAR(128)	Nombre de la columna espacial que deben mantener los activadores que este procedimiento almacenado crea o vuelve a activar. Este parámetro no se puede anular.

Tabla 11. Parámetros de entrada para el procedimiento almacenado db2gse.gse_enable_autogc. (continuación)

Nombre	Tipo de datos	Descripción
gcId	INTEGER	Identificador del geocodificador que invocarán los activadores de inserción y actualización que este procedimiento almacenado crea o vuelve a activar. Este parámetro no se puede anular si el valor del parámetro operMode es GSE_AUTOGC_CREATE. Se puede anular si el valor de operMode es GSE_AUTOGC_RECREATE.
precisionLevel	INTEGER	El grado en que los datos fuente deben coincidir con los datos de referencia correspondientes para que el geocodificador procese satisfactoriamente los datos fuente. Este parámetro no se puede anular si el valor del parámetro operMode es GSE_AUTOGC_CREATE. Se puede anular si el valor de operMode es GSE_AUTOGC_RECREATE. **Comentario:** El nivel de precisión puede estar comprendido entre 1 y 100 por ciento.
vendorSpecific	VARCHAR(256)	Información técnica suministrada por el proveedor; por ejemplo, la vía de acceso y el nombre de un archivo que utiliza el proveedor para definir parámetros. Este parámetro no se puede anular si el valor del parámetro operMode es GSE_AUTOGC_CREATE. Se puede anular si el valor de operMode es GSE_AUTOGC_RECREATE.

Parámetros de salida

Tabla 12. Parámetros de salida para el procedimiento almacenado db2gse.gse_enable_autogc.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

Restricciones

- El parámetro layerColumn debe hacer referencia a una columna registrada como una capa de tabla.
- Si el valor del parámetro operMode es GSE_AUTOGC_CREATE, debe asignar un identificador de un geocodificador registrado al parámetro gcId.

db2gse.gse_enable_db

Utilice este procedimiento almacenado para suministrar a una base de datos los recursos que necesita para almacenar datos espaciales y para dar soporte a operaciones. Estos recursos incluyen tipos de datos espaciales, un tipo de índice espacial, vistas y tablas de catálogo, funciones suministradas y otros procedimientos almacenados.

Para ver un ejemplo del código que sirve para invocar este procedimiento almacenado, consulte la función C en el programa de ejemplo. Para obtener información sobre este programa, consulte el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59.

Autorización

El ID de usuario bajo el que se invoca el procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos que se está habilitando.

Parámetros de salida

Tabla 13. Parámetros de salida para el procedimiento almacenado db2gse.gse_enable_db.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

db2gse.gse_enable_idx

Utilice este procedimiento almacenado para crear un índice para una columna espacial.

Para ver un ejemplo del código que sirve para invocar este procedimiento almacenado, consulte la función C gseEnableIdx en el programa de ejemplo. Para obtener información sobre este programa, consulte el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59.

Autorización

El ID de usuario bajo el que se invoca este procedimiento almacenado debe tener uno de los siguientes privilegios o autorizaciones:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla para la que se va a utilizar el índice habilitado.
- El privilegio CONTROL o INDEX sobre esta tabla.

Parámetros de entrada

Tabla 14. Parámetros de entrada para el procedimiento almacenado db2gse.gse_enable_idx.

Nombre	Tipo de datos	Descripción
layerSchema	VARCHAR (30)	Nombre del esquema al que pertenece la tabla especificada en el parámetro layerTable. Este parámetro no se puede anular. **Comentario:** Si no especifica un valor para el parámetro layerSchema, este adoptará como valor por omisión el ID de usuario bajo el que se invoca el procedimiento almacenado db2gse.gse_enable_idx.
layerTable	VARCHAR(128)	Nombre de la tabla en la que se va a definir el índice que está creando. Este parámetro no se puede anular.
layerColumn	VARCHAR(128)	Nombre de la columna habilitada para operaciones espaciales que se va a buscar con la ayuda del índice que está creando. Este parámetro no se puede anular.

Tabla 14. Parámetros de entrada para el procedimiento almacenado db2gse.gse_enable_idx. (continuación)

Nombre	Tipo de datos	Descripción
indexName	VARCHAR(128)	Nombre del índice que se va a crear. Este parámetro no se puede anular. **Comentario:** No especifique ningún nombre de esquema. DB2 Spatial Extender asigna automáticamente el índice al esquema al que se hacer referencia en el parámetro layerSchema.
gridSize1	DOUBLE	Número que indica la granularidad que debe tener la cuadrícula de índice más fina. Este parámetro no se puede anular.
gridSize2	DOUBLE	Número que indica (1) que no debe haber una segunda cuadrícula para este índice o (2) cuál debe ser la granularidad de la segunda cuadrícula. Este parámetro no se puede anular. **Comentario:** Si no debe haber una segunda cuadrícula de índice, especifique 0. Si desea una segunda cuadrícula, su granularidad debe ser menor que la indicada por gridSize1.
gridSize3	DOUBLE	Número que indica (1) que no debe haber una tercera cuadrícula para este índice o (2) cuál debe ser la granularidad de la tercera cuadrícula. Este parámetro no se puede anular. **Comentario:** Si no debe haber una tercera cuadrícula de índice, especifique 0. Si desea una tercera cuadrícula, su granularidad debe ser menor que la indicada por gridSize2.

Parámetros de salida

Tabla 15. Parámetros de salida para el procedimiento almacenado db2gse.gse_enable_idx.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

db2gse.gse_enable_sref

Utilice este procedimiento almacenado para especificar cómo se deben convertir los números negativos y decimales de un sistema de coordenadas especificado en enteros positivos para que DB2 Spatial Extender los pueda almacenar. Sus especificaciones se denominan en conjunto *sistema de referencias espaciales*. Cuando se procesa este procedimiento almacenado, se añade información sobre el sistema de referencias espaciales a la vista de catálogo DB2GSE.SPATIAL_REF_SYS. Para obtener información sobre esta vista, consulte el tema “DB2GSE.SPATIAL_REF_SYS” en la página 120.

Para ver un ejemplo del código que sirve para invocar este procedimiento almacenado, consulte la función C gseEnableSref en el programa de ejemplo. Para obtener información sobre este programa, consulte el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59.

Autorización

No se necesita.

Parámetros de entrada

Tabla 16. Parámetros de entrada para el procedimiento almacenado db2gse.gse_enable_sref.

Nombre	Tipo de datos	Descripción
srId	INTEGER	Un identificador numérico para el sistema de referencias espaciales. Este parámetro no se puede anular. **Comentario**: Este identificador debe ser exclusivo dentro de su base de datos habilitada para operaciones espaciales.
srName	VARCHAR(64)	Breve descripción del sistema de referencias espaciales. Este parámetro no se puede anular. **Comentario**: Esta descripción debe ser exclusiva dentro de su base de datos habilitada para operaciones espaciales.
falsex	DOUBLE	Un número que, al restarse de un valor negativo de coordenada X, da como resultado un número no negativo (es decir, un número positivo o un cero). Este parámetro no se puede anular.

Tabla 16. Parámetros de entrada para el procedimiento almacenado db2gse.gse_enable_sref. (continuación)

Nombre	Tipo de datos	Descripción
falsey	DOUBLE	Un número que, al restarse de un valor negativo de coordenada Y, da como resultado un número no negativo (es decir, un número positivo o un cero). Este parámetro no se puede anular.
xyunits	DOUBLE	Un número que, cuando se multiplica por una coordenada X decimal o por una coordenada Y decimal, da como resultado un entero que se puede almacenar como un elemento de datos de 32 bits. Este parámetro no se puede anular.
falsez	DOUBLE	Un número que, al restarse de un valor negativo de coordenada Z, da como resultado un número no negativo (es decir, un número positivo o un cero). Este parámetro no se puede anular.
zunits	DOUBLE	Un número que, cuando se multiplica por una coordenada Z decimal, da como resultado un entero que se puede almacenar como un elemento de datos de 32 bits. Este parámetro no se puede anular.
falsem	DOUBLE	Un número que, al restarse de una medida negativa, da como resultado un número no negativo (es decir, un número positivo o un cero). Este parámetro no se puede anular.
munits	DOUBLE	Un número que, cuando se multiplica por una medida decimal, da como resultado un entero que se puede almacenar como un elemento de datos de 32 bits. Este parámetro no se puede anular.

Tabla 16. Parámetros de entrada para el procedimiento almacenado db2gse.gse_enable_sref. (continuación)

Nombre	Tipo de datos	Descripción
scId	INTEGER	Identificador numérico del sistema de coordenadas a partir del cual se obtiene el sistema de referencias espaciales. Para saber qué es un identificador numérico del sistema de coordenadas, consulte la vista de catálogo DB2GSE.COORD_REF_SYS en "DB2GSE.COORD_REF_SYS" en la página 117. Este parámetro no se puede anular.

Parámetros de salida

Tabla 17. Parámetros de salida para el procedimiento almacenado db2gse.gse_enable_sref.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

db2gse.gse_export_shape

Utilice este procedimiento almacenado para exportar una capa y su tabla asociada a un archivo de forma o para crear un nuevo archivo de forma y exportar una capa y su tabla asociada a este nuevo archivo.

Para ver un ejemplo del código que sirve para invocar este procedimiento almacenado, consulte la función C gseExportShape en el programa de ejemplo. Para obtener información sobre este programa, consulte el "Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 59.

Autorización

El ID de usuario bajo el cual se invoca este procedimiento almacenado debe tener el privilegio SELECT sobre la tabla que se va a exportar.

Parámetros de entrada

Tabla 18. Parámetros de entrada para el procedimiento almacenado db2gse.gse_export_shape.

Nombre	Tipo de datos	Descripción
layerSchema	VARCHAR (30)	Nombre del esquema al que pertenece la tabla especificada en el parámetro layerTable. Este parámetro no se puede anular. **Comentario:** Si no especifica un valor para el parámetro layerSchema, este adoptará como valor por omisión el ID de usuario bajo el que se invoca el procedimiento almacenado db2gse.gse_export_shape.
layerTable	VARCHAR(128)	Nombre de la tabla que está exportando. Este parámetro no se puede anular.
layerColumn	VARCHAR (30)	Nombre de la columna que se ha registrado como la capa que está exportando. Este parámetro no se puede anular.
fileName	VARCHAR(128)	Nombre del archivo de forma al que se va a exportar la capa especificada. Este parámetro no se puede anular.

Tabla 18. Parámetros de entrada para el procedimiento almacenado db2gse.gse_export_shape. (continuación)

Nombre	Tipo de datos	Descripción
whereClause	VARCHAR(1024)	La parte central de la cláusula SQL WHERE. Define una restricción sobre el grupo de registros que se deben geocodificar. La cláusula puede hacer referencia a cualquier columna de atributo de la tabla que está exportando. Este parámetro no se puede anular.

Parámetros de salida

Tabla 19. Parámetros de salida para el procedimiento almacenado db2gse.gse_export_shape.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

Restricción

No puede exportar más de una capa simultáneamente.

db2gse.gse_import_sde

Utilice este procedimiento almacenado para importar un archivo de transferencia SDE a una base de datos que se ha habilitado para operaciones espaciales. El procedimiento almacenado puede funcionar de dos formas:

- Si el archivo de transferencia SDE va destinado a una tabla existente que tiene una columna de capa registrada, DB2 Spatial Extender cargará la tabla con los datos del archivo.
- Si no es así, DB2 Spatial Extender creará una tabla con una columna espacial, registrará esta columna como una capa y cargará la capa y las otras columnas de la tabla con los datos del archivo.

El sistema de referencias espaciales especificado en el archivo de transferencia SDE se comparará con los sistemas de referencias espaciales registrados ante DB2 Spatial Extender. Si el sistema especificado coincide con un sistema registrado, los valores negativos y decimales de los datos de trasferencia se modificarán, cuando se carguen, tal como indique el sistema registrado. Si el sistema especificado no coincide con ninguno de los sistemas registrados, DB2 Spatial Extender creará un nuevo sistema de referencias espaciales para indicar las modificaciones.

Autorización

Cuando importa datos a una tabla existente, el ID de usuario bajo el que se invoca este procedimiento almacenado debe tener una de las siguientes autorizaciones o privilegios:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla en la que se deben importar los datos.
- El privilegio CONTROL sobre esta tabla.

Cuando la tabla en la que desea importar datos se debe crear, el ID de usuario bajo el que se invoca este procedimiento almacenado debe tener una de las siguientes autorizaciones o privilegios.

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla que se va a crear.

Parámetros de entrada

Tabla 20. Parámetros de entrada para el procedimiento almacenado db2gse.gse_import_sde.

Nombre	Tipo de datos	Descripción
layerSchema	VARCHAR (30)	Nombre del esquema al que pertenece la tabla o vista especificada en el parámetro layerTable. Este parámetro no se puede anular. **Comentario:** Si no especifica un valor para el parámetro layerSchema, este adoptará como valor por omisión el ID de usuario bajo el que se invoca el procedimiento almacenado db2gse.gse_import_sde.
layerTable	VARCHAR(128)	Nombre de la tabla en la que se deben cargar los datos de transferencia SDE. Este parámetro no se puede anular.
layerColumn	VARCHAR (30)	Nombre de la columna que se ha registrado como la capa en la que se deben cargar los datos espaciales del archivo de transferencia SDE. Este parámetro no se puede anular.
fileName	VARCHAR(128)	Nombre del archivo de transferencia SDE que se debe importar. Este parámetro no se puede anular.
commitScope	INTEGER	Número de registros por punto de control. Este parámetro no se puede anular.

Parámetros de salida

Tabla 21. Parámetros de salida para el procedimiento almacenado db2gse.gse_import_sde.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

db2gse.gse_import_shape

Utilice este procedimiento almacenado para importar un archivo de forma a una base de datos que se ha habilitado para operaciones espaciales. El procedimiento almacenado puede funcionar de dos formas:

- Si el archivo de forma va destinado a una tabla existente que tiene una columna de capa registrada, DB2 Spatial Extender cargará la tabla con los datos del archivo.
- Si no es así, DB2 Spatial Extender creará una tabla con una columna espacial, registrará esta columna como una capa y cargará la capa y las otras columnas de la tabla con los datos del archivo.

Para ver un ejemplo del código que sirve para invocar este procedimiento almacenado, consulte la función C gseImportShape en el programa de ejemplo. Para obtener información sobre este programa, consulte el "Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 59.

Autorización

El ID de usuario bajo el que se invoca este procedimiento almacenado debe tener uno de los siguientes privilegios o autorizaciones:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla en la que se van a cargar los datos de forma importados.
- El privilegio CONTROL sobre esta tabla.

Parámetros de entrada

Tabla 22. Parámetros de entrada para el procedimiento almacenado db2gse.gse_import_shape.

Nombre	Tipo de datos	Descripción
layerSchema	VARCHAR (30)	Nombre del esquema al que pertenece la tabla o vista especificada en el parámetro layerTable. Este parámetro no se puede anular. **Comentario:** Si no especifica un valor para el parámetro layerSchema, este adoptará como valor por omisión el ID de usuario bajo el que se invoca el procedimiento almacenado db2gse.gse_import_shape.
layerTable	VARCHAR(128)	Nombre de la tabla en la que se va a cargar el archivo de forma importado. Este parámetro no se puede anular.

Tabla 22. Parámetros de entrada para el procedimiento almacenado db2gse.gse_import_shape. (continuación)

Nombre	Tipo de datos	Descripción
layerColumn	VARCHAR (30)	Nombre de la columna que se ha registrado como la capa en la que se van a cargar los datos de forma. Este parámetro no se puede anular.
fileName	VARCHAR(128)	Nombre del archivo de forma que se va a importar. Este parámetro no se puede anular.
exceptionFile	VARCHAR(128)	Vía de acceso y nombre del archivo en el que se almacenan las formas que no se han podido importar. Es un archivo nuevo que se creará cuando se ejecute el procedimiento almacenado db2gse.gse_import_shape. Este parámetro no se puede anular.
srId	INTEGER	Identificador del sistema de referencias espaciales que se debe utilizar para la capa en la que se van a cargar los datos de forma. Este parámetro no se puede anular. **Comentario:** Si no se especifica este identificador, la transformación interna se definirá con la máxima resolución posible para el archivo de forma.
commitScope	INTEGER	Número de registros por punto de control. Este parámetro no se puede anular.

Parámetros de salida

Tabla 23. Parámetros de salida para el procedimiento almacenado db2gse.gse_import_shape.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

db2gse.gse_register_gc

Utilice este procedimiento almacenado para registrar un geocodificador que no sea el geocodificador por omisión. Para saber si un geocodificador ya se ha registrado, consulte la vista de catálogo DB2GSE.SPATIAL_GEOCODER (descrita en el tema “DB2GSE.SPATIAL_GEOCODER” en la página 119).

Autorización

El ID de usuario bajo el que se invoca este procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos que contiene el geocodificador que registra este procedimiento almacenado.

Parámetros de entrada

Tabla 24. Parámetros de entrada para el procedimiento almacenado db2gse.gse_register_gc.

Nombre	Tipo de datos	Descripción
gcId	INTEGER	Identificador numérico del geocodificador que está registrando. Este parámetro no se puede anular. **Comentario:** Este identificador debe ser exclusivo dentro de la base de datos.
gcName	VARCHAR(64)	Breve descripción del geocodificador que está registrando. Este parámetro no se puede anular. **Comentario:** Esta descripción debe ser una serie de caracteres exclusiva dentro de la base de datos.
vendorName	VARCHAR(64)	Nombre del proveedor que ha suministrado el geocodificador que está registrando. Este parámetro no se puede anular.
primaryUDF	VARCHAR(256)	Nombre calificado al completo del geocodificador que está registrando. Este parámetro no se puede anular.
precisionLevel	INTEGER	El grado en que los datos fuente deben coincidir con los datos de referencia correspondientes para que el geocodificador procese satisfactoriamente los datos fuente. Este parámetro no se puede anular. **Comentario:** El nivel de precisión puede estar comprendido entre 1 y 100 por ciento.

Tabla 24. Parámetros de entrada para el procedimiento almacenado db2gse.gse_register_gc. (continuación)

Nombre	Tipo de datos	Descripción
vendorSpecific	VARCHAR(256)	Información técnica suministrada por el proveedor; por ejemplo, la vía de acceso y el nombre de un archivo que utiliza el proveedor para definir parámetros. Este parámetro no se puede anular.
geoArea	VARCHAR(256)	Área geográfica que se debe geocodificar. Este parámetro no se puede anular.
descripción	VARCHAR(256)	Observaciones suministradas por el proveedor. Este parámetro no se puede anular.

Parámetros de salida

Tabla 25. Parámetros de salida para el procedimiento almacenado db2gse.gse_register_gc.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

db2gse.gse_register_layer

Utilice este procedimiento almacenado para registrar una columna espacial como una capa. Cuando se procesa este procedimiento almacenado, se añade información sobre la capa que se registra a la vista de catálogo DB2GSE.GEOMETRY_COLUMNS. Para obtener información sobre esta vista, consulte el tema “DB2GSE.GEOMETRY_COLUMNS” en la página 118.

Para ver un ejemplo del código que sirve para invocar este procedimiento almacenado, consulte la función C gseRegisterLayer en el programa de ejemplo. Para obtener información sobre este programa, consulte el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59.

Autorización

El ID de usuario bajo el que se invoca este procedimiento almacenado debe tener uno de los siguientes privilegios o autorizaciones:

- Para una capa de tabla:
  - Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla a la que pertenece esta capa.
  - El privilegio CONTROL o ALTER sobre esta tabla.
- Para una capa de vista:
  - El privilegio SELECT sobre la tabla o tablas base que contienen (1) los datos de dirección que se deben geocodificar para esta capa y (2) los datos espaciales resultantes de la geocodificación.

Parámetros de entrada

Tabla 26. Parámetros de entrada para el procedimiento almacenado db2gse.gse_register_layer.

Nombre	Tipo de datos	Descripción
layerSchema	INTEGER(30)	Nombre del esquema al que pertenece la tabla o vista especificada en el parámetro layerTable. Este parámetro no se puede anular. **Comentario:** Si no especifica un valor para el parámetro layerSchema, este adoptará como valor por omisión el ID de usuario bajo el que se invoca el procedimiento almacenado db2gse.gse_register_layer.
layerTable	VARCHAR(128)	Nombre de la tabla o vista que contiene la columna que se registra como una capa. Este parámetro no se puede anular.

Tabla 26. Parámetros de entrada para el procedimiento almacenado db2gse.gse_register_layer. (continuación)

Nombre	Tipo de datos	Descripción
layerColumn	VARCHAR(128)	Nombre de la columna que se está registrando como una capa. Si la columna no existe, DB2 Spatial Extender la creará. Este parámetro no se puede anular.
layerTypeName	VARCHAR(64)	Tipo de datos de la columna que se está registrando como una capa. Debe especificar el tipo de datos en mayúsculas; por ejemplo: `ST_POINT` Este parámetro no se puede anular si la columna es una columna de tabla que se debe crear cuando se procese este procedimiento almacenado. De lo contrario, si la columna es una columna existente dentro de una tabla o vista, este parámetro se puede anular.
srId	INTEGER	Identificador del sistema de referencias espaciales utilizado para esta capa. Este parámetro no se puede anular para una capa de tabla. DB2 Spatial Extender pasa por alto este parámetro cuando el usuario registra una capa de vista.
geoSchema	VARCHAR (30)	Se aplica cuando el usuario registra una columna de vista como una capa. El parámetro geoSchema es el esquema de la tabla subyacente a la vista a la que pertenece la columna. Este parámetro se puede anular cuando registra una columna de vista como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de tabla como una capa. **Comentario:** Si no especifica un valor para el parámetro geoSchema, este adoptará como valor por omisión el valor del parámetro layerSchema.

Tabla 26. Parámetros de entrada para el procedimiento almacenado db2gse.gse_register_layer. (continuación)

Nombre	Tipo de datos	Descripción
geoTable	VARCHAR(128)	Se aplica cuando el usuario registra una columna de vista como una capa. El parámetro geoTable es el nombre de la tabla subyacente a la vista a la que pertenece la columna. Este parámetro no se puede anular cuando registra una columna de vista como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de tabla como una capa.
geoColumn	VARCHAR(128)	Se aplica cuando el usuario registra una columna de vista como una capa. El parámetro geoColumn es el nombre de la columna de tabla subyacente a esta columna de vista. Este parámetro no se puede anular cuando registra una columna de vista como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de tabla como una capa.
nAttributes	SMALLINT	Número de columnas que contienen los datos fuente que se deben geocodificar para esta capa. Este parámetro se puede anular cuando registra una columna de tabla como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de vista como una capa.
attr1Name	VARCHAR(128)	Nombre de la primera columna que contiene los datos fuente que se deben geocodificar para esta capa. Este parámetro se puede anular cuando registra una columna de tabla como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de vista como una capa. Si tiene intención de utilizar el geocodificador por omisión, tiene que guardar las direcciones de calles en la columna attr1Name.

Tabla 26. Parámetros de entrada para el procedimiento almacenado db2gse.gse_register_layer. (continuación)

Nombre	Tipo de datos	Descripción
attr2Name	VARCHAR(128)	Nombre de la segunda columna que contiene los datos fuente que se deben geocodificar para esta capa. Este parámetro se puede anular cuando registra una columna de tabla como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de vista como una capa. Si tiene intención de utilizar el geocodificador por omisión, tiene que guardar los nombres de ciudades en la columna attr2Name.
attr3Name	VARCHAR(128)	Nombre de la tercera columna que contiene los datos fuente que se deben geocodificar para esta capa. Este parámetro se puede anular cuando registra una columna de tabla como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de vista como una capa. Si tiene intención de utilizar el geocodificador por omisión, tiene que guardar los nombres o abreviaturas de estados en la columna attr3Name.
attr4Name	VARCHAR(128)	Nombre de la cuarta columna que contiene los datos fuente que se deben geocodificar para esta capa. Este parámetro se puede anular cuando registra una columna de tabla como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de vista como una capa. Si tiene intención de utilizar el geocodificador por omisión, tiene que guardar los códigos postales en la columna attr4Name.

Tabla 26. Parámetros de entrada para el procedimiento almacenado db2gse.gse_register_layer. (continuación)

Nombre	Tipo de datos	Descripción
attr5Name	VARCHAR(128)	Nombre de la quinta columna que contiene los datos fuente que se deben geocodificar para esta capa. Este parámetro se puede anular cuando registra una columna de tabla como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de vista como una capa. El geocodificador por omisión pasa por alto la columna Attr5Name.
attr6Name	VARCHAR(128)	Nombre de la sexta columna que contiene los datos fuente que se deben geocodificar para esta capa. Este parámetro se puede anular cuando registra una columna de tabla como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de vista como una capa. El geocodificador por omisión pasa por alto la columna Attr6Name.
attr7Name	VARCHAR(128)	Nombre de la séptima columna que contiene los datos fuente que se deben geocodificar para esta capa. Este parámetro se puede anular cuando registra una columna de tabla como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de vista como una capa. El geocodificador por omisión pasa por alto la columna Attr7Name.

Tabla 26. Parámetros de entrada para el procedimiento almacenado db2gse.gse_register_layer. (continuación)

Nombre	Tipo de datos	Descripción
attr8Name	VARCHAR(128)	Nombre de la octava columna que contiene los datos fuente que se deben geocodificar para esta capa. Este parámetro se puede anular cuando registra una columna de tabla como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de vista como una capa. El geocodificador por omisión pasa por alto la columna Attr8Name.
attr9Name	VARCHAR(128)	Nombre de la novena columna que contiene los datos fuente que se deben geocodificar para esta capa. Este parámetro se puede anular cuando registra una columna de tabla como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de vista como una capa. El geocodificador por omisión pasa por alto la columna Attr9Name.
attr10Name	VARCHAR(128)	Nombre de la décima columna que contiene los datos fuente que se deben geocodificar para esta capa. Este parámetro se puede anular cuando registra una columna de tabla como una capa. DB2 Spatial Extender pasa por alto este parámetro cuando registra una columna de vista como una capa. El geocodificador por omisión pasa por alto la columna Attr10Name.

Parámetros de salida

Tabla 27. Parámetros de salida para el procedimiento almacenado db2gse.gse_register_layer.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

Restricciones

- Si está registrando una columna de vista como una capa, esta se debe basar en una columna de tabla que ya se haya registrado como una capa.
- No puede haber más de diez columnas de atributo que contengan los datos que se deben geocodificar para la capa que está registrando.

db2gse.gse_run_gc

Utilice este procedimiento almacenado para ejecutar un geocodificador en modalidad de proceso por lotes. Para obtener información sobre esta tarea, consulte el tema “Ejecución del geocodificador en modalidad de proceso por lotes” en la página 44.

Para ver un ejemplo del código que sirve para invocar este procedimiento almacenado, consulte la función C gseRunGC en el programa de ejemplo. Para obtener información sobre este programa, consulte el “Capítulo 8. Cómo escribir aplicaciones para DB2 Spatial Extender” en la página 59.

Autorización

El ID de usuario bajo el que se invoca este procedimiento almacenado debe tener uno de los siguientes privilegios o autorizaciones:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla sobre la que va a operar el geocodificador especificado.
- El privilegio CONTROL o UPDATE sobre esta tabla.

Parámetros de entrada

Tabla 28. Parámetros de entrada para el procedimiento almacenado db2gse.gse_run_gc.

Nombre	Tipo de datos	Descripción
layerSchema	VARCHAR (30)	Nombre del esquema al que pertenece la tabla o vista especificada en el parámetro layerTable. Este parámetro no se puede anular. **Comentario:** Si no especifica un valor para el parámetro layerSchema, este adoptará como valor por omisión el ID de usuario bajo el que se invoca el procedimiento almacenado db2gse.gse_run_gc.
layerTable	VARCHAR(128)	Nombre de la tabla que contiene la columna en la que se van a insertar los datos geocodificados. Este parámetro no se puede anular.
layerColumn	VARCHAR(128)	Nombre de la columna en la que se van a insertar los datos geocodificados. Este parámetro no se puede anular.

Tabla 28. Parámetros de entrada para el procedimiento almacenado db2gse.gse_run_gc. (continuación)

Nombre	Tipo de datos	Descripción
gcId	INTEGER	Identifica el geocodificador que desea ejecutar. Este parámetro no se puede anular. Para buscar los identificadores de geocodificadores registrados, consulte la vista de catálogo DB2GSE.SPATIAL_GEOCODER.
precisionLevel	INTEGER	El grado en que los datos fuente deben coincidir con los datos de referencia correspondientes para que el geocodificador procese satisfactoriamente los datos fuente. Este parámetro no se puede anular. **Comentario:** El nivel de precisión puede estar comprendido entre 1 y 100 por ciento.
vendorSpecific	VARCHAR(256)	Información técnica suministrada por el proveedor; por ejemplo, la vía de acceso y el nombre de un archivo que utiliza el proveedor para definir parámetros. Este parámetro no se puede anular.
whereClause	VARCHAR(256)	La parte central de la cláusula WHERE. Define una restricción sobre el grupo de registros que se deben geocodificar. La cláusula puede hacer referencia a cualquier columna de atributo de la tabla sobre la que opera el geocodificador. Este parámetro no se puede anular.
commitScope	INTEGER	Número de registros por punto de control. Este parámetro no se puede anular.

Parámetros de salida

Tabla 29. Parámetros de salida para el procedimiento almacenado db2gse.gse_run_gc.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

db2gse.gse_unregist_gc

Utilice este procedimiento almacenado para desregistrar un geocodificador que no sea el geocodificador por omisión.

Para buscar información sobre el geocodificador que desea desregistrar, consulte la vista de catálogo DB2GSE.SPATIAL_GEOCODER, consulte el tema “DB2GSE.SPATIAL_GEOCODER” en la página 119.

Autorización

El ID de usuario bajo el que se invoca este procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos que contiene el geocodificador que se va a desregistrar.

Parámetro de entrada

Tabla 30. Parámetros de entrada para el procedimiento almacenado db2gse.gse_unregist_gc.

Nombre	Tipo de datos	Descripción
gcId	INTEGER	El identificador del geocodificador que se va a desregistrar. Este parámetro no se puede anular.

Parámetros de salida

Tabla 31. Parámetros de salida para el procedimiento almacenado db2gse.gse_unregist_gc.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

db2gse.gse_unregist_layer

Utilice este procedimiento almacenado para desregistrar una capa. Para ello, el procedimiento almacenado:

- Elimina la definición de la capa de las tablas de catálogo de DB2 Spatial Extender.
- Suprime la restricción de control que DB2 Spatial Extender ha colocado en la tabla base de esta capa para asegurar que los datos espaciales de la capa cumplen con los requisitos del sistema de referencias espaciales de la capa.
- Elimina los activadores que se utilizan para activar la columna espacial cuando se añaden, modifican o eliminan datos de dirección.

Cuando se procesa el procedimiento almacenado, se elimina información sobre la capa de la meta vista DB2GSE.GEOMETRY_COLUMNS. Para obtener información sobre esta vista, consulte el tema "DB2GSE.GEOMETRY_COLUMNS" en la página 118.

Autorización

El ID de usuario bajo el que se invoca este procedimiento almacenado debe tener uno de los siguientes privilegios o autorizaciones:

- Para una capa de tabla:
  - Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla base de esta capa.
  - El privilegio CONTROL o ALTER sobre esta tabla.
- Para una capa de vista:
  - El privilegio SELECT sobre la tabla o tablas base que contienen (1) los datos de dirección que se geocodifican para esta capa y (2) los datos espaciales resultantes de la geocodificación.

Parámetros de entrada

Tabla 32. Parámetros de entrada para el procedimiento almacenado db2gse.gse_unregist_layer.

Nombre	Tipo de datos	Descripción
layerSchema	VARCHAR (30)	Nombre del esquema al que pertenece la tabla especificada en el parámetro layerTable. Este parámetro no se puede anular. **Comentario:** Si no especifica un valor para el parámetro layerSchema, este adoptará como valor por omisión el ID de usuario bajo el que se invoca el procedimiento almacenado db2gse.gse_unregister_layer.

Tabla 32. Parámetros de entrada para el procedimiento almacenado db2gse.gse_unregist_layer. (continuación)

Nombre	Tipo de datos	Descripción
layerTable	VARCHAR(128)	Nombre de la tabla que contiene la columna especificada en el parámetro layerColumn. Este parámetro no se puede anular.
layerColumn	VARCHAR(128)	Nombre de la columna espacial que se ha definido como la capa que desea desregistrar. Este parámetro no se puede anular.

Parámetros de salida

Tabla 33. Parámetros de salida para el procedimiento almacenado db2gse.gse_unregist_layer.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que puede devolver el emisor de la llamada a este procedimiento almacenado.
Reserved	VARCHAR(1024)	Mensaje de error completo, tal como está definido en el servidor de DB2 Spatial Extender.

Restricción

Si una columna de vista definida como una capa de vista se basa en una columna de tabla definida como una capa de tabla, no puede desregistrar esta capa de tabla hasta que desregistre la capa de vista.

Capítulo 10. Mensajes

Este capítulo documenta los mensajes que DB2 Spatial Extender devuelve a los usuarios. Cada mensaje tiene un identificador. Los identificadores que terminan por la letra E corresponden a mensajes de error, los que terminan por W corresponden a avisos y los que terminan por I corresponden a mensajes de información general.

DBA7200E Se han seleccionado más de 10 columnas como entrada de un geocodificador.

Explicación: Se pueden seleccionar un máximo de 10 columnas como entrada de un geocodificador.

Respuesta del usuario: Mueva nombres de columnas del recuadro Columnas seleccionadas al recuadro Columnas disponibles hasta que el recuadro Columnas seleccionadas contenga diez nombres o menos.

DBA7201E La base de datos no está habilitada para el funcionamiento de Spatial Extender.

Explicación: La base de datos debe estar habilitada para Spatial Extender para que pueda utilizar Spatial Extender.

Respuesta del usuario: Pulse con el botón derecho del ratón la base de datos y seleccione Spatial Extender —> Habilitar en los menús.

GSE0000I La operación ha finalizado satisfactoriamente.

GSE0001E Spatial Extender no ha podido realizar la operación solicitada (“<nombre-operación>”) bajo el ID de usuario “<id-usuario>”.

Explicación: Ha solicitado esta operación bajo un ID de usuario que no tiene el privilegio o la autorización necesarios para llevar a cabo la operación.

Respuesta del usuario: Consulte la documentación para saber qué autorización se necesita u obténgala de un administrador de Spatial Extender.

GSE0002E “<valor>” no es un valor válido para el argumento “<nombre-argumento>”.

Explicación: El valor que ha especificado es incorrecto o está mal escrito.

Respuesta del usuario: Consulte la documentación o consulte con un administrador de Spatial Extender para saber qué valor o rango de valores tiene que especificar.

GSE0003E Spatial Extender no ha podido llevar a cabo la operación solicitada porque no se ha especificado el argumento “<nombre-argumento>”.

Explicación: No ha especificado un argumento necesario para esta operación.

Respuesta del usuario: Especifique el argumento “<nombre-argumento>” con el valor que desee; luego vuelva a solicitar la operación.

GSE0004W El argumento “<nombre-argumento>” no se ha evaluado.

Explicación: La operación que ha solicitado no utiliza el argumento “<nombre-argumento>”.

Respuesta del usuario: No se necesita.

GSE0005E Spatial Extender no ha podido procesar su solicitud de crear un objeto denominado "<nombre-objeto>".

Explicación: El objeto "<nombre-objeto>" ya existe o no tiene el permiso adecuado para crearlo. Puede tratarse de una tabla, de una columna, de un activador, de un índice o de otro tipo de objeto.

Respuesta del usuario: Si "<nombre-objeto>" es el objeto que desea, no haga nada. Si no es así, especifique el nombre correctamente y compruebe que tiene el permiso adecuado para crear el objeto.

GSE0006E Spatial Extender no ha podido llevar a cabo la operación solicitada sobre un objeto habilitado o registrado denominado "<nombre-objeto>".

Explicación: El objeto "<nombre-objeto>" ya está habilitado o registrado, o bien ya existe. Puede tratarse de una capa, de un índice, de un sistema de referencias espaciales, de un sistema de coordenadas, de un geocodificador o de otro tipo de objeto.

Respuesta del usuario: Asegúrese de que el objeto "<nombre-objeto>" existe y vuelva a someter su solicitud.

GSE0007E Spatial Extender no ha podido llevar a cabo la operación solicitada sobre "<nombre-objeto>", un objeto que aún no se ha habilitado o registrado.

Explicación: El objeto "<nombre-objeto>" no se ha habilitado o registrado. Puede tratarse de una capa, de un índice, de un sistema de referencias espaciales, de un sistema de coordenadas espaciales, de un geocodificador o de otro tipo de objeto.

Respuesta del usuario: Habilite o registre el objeto "<nombre-objeto>". Luego vuelva a someter su solicitud.

GSE0008E Se ha producido un error inesperado de SQL ("<mensaje-error-sql>").

Respuesta del usuario: Consulte el mensaje detallado asociado a SQLCODE en el mensaje de error de SQL "<mensaje-error-sql>". Si es necesario, póngase en contacto con el representante de servicio de IBM.

GSE0009E No se ha podido llevar a cabo la operación solicitada sobre un objeto denominado "<nombre-objeto>" que ya existe.

Explicación: "<nombre-objeto>" ya existe en la base de datos o en el sistema operativo. Puede tratarse de un archivo, de una tabla, de una vista, de una columna, de un índice, de un activador o de otro tipo de objeto.

Respuesta del usuario: Asegúrese de especificar el objeto correctamente cuando intente acceder al mismo. Si es necesario, suprima el objeto.

GSE0010E No se ha podido llevar a cabo la operación solicitada sobre un objeto denominado "<nombre-objeto>" que es posible que no exista.

Explicación: "<nombre-objeto>" no existe en la base de datos o en el sistema operativo. Puede tratarse de un archivo, de una tabla, de una vista, de una columna, de un índice, de un activador, de un archivo o de otro tipo de objeto.

Respuesta del usuario: Asegúrese de que tiene el permiso adecuado para acceder al objeto. Si tiene este permiso y el objeto no existe, tendrá que crearlo.

GSE0011E Spatial Extender no ha podido inhabilitar o desregistrar el objeto "<nombre-objeto>".

Explicación: "<nombre-objeto>" depende de otro objeto. "<nombre-objeto>" puede ser un sistema de referencias espaciales, una capa, un geocodificador u otro tipo de objeto.

Respuesta del usuario: Consulte la documentación para ver de qué tipos de objetos puede depender "<nombre-objeto>". Luego elimine el objeto específico del que depende "<nombre-objeto>".

GSE0012E **Spatial Extender no ha podido procesar su solicitud porque la columna espacial calificada al completo "<esquema-capa.nombre-capa.columna-capa>" no está registrada como una capa de tabla.**

Explicación: La columna espacial calificada al completo "<esquema-capa.nombre-capa.columna-capa>" debe estar registrada como una capa de tabla para que pueda realizar ciertas operaciones asociadas a la misma (por ejemplo, habilitar su índice, habilitar un geocodificador para llenarla en modalidad de proceso por lotes o para actualizarla automáticamente).

Respuesta del usuario: Asegúrese de que la columna espacial calificada al completo "<esquema-capa.nombre-capa.columna-capa>" está registrada como una capa de tabla comprobando la vista DB2GSE.GEOMETRY_COLUMNS en el catálogo de Spatial Extender. Asegúrese también de que la tabla que contiene esta columna incluye también columnas de atributo correspondientes válidas.

GSE0013E **La base de datos no está habilitada para operaciones espaciales.**

Explicación: La base de datos no está habilitada para operaciones espaciales. Por lo tanto, el catálogo de Spatial Extender no existe.

Respuesta del usuario: Habilite la base de datos para operaciones espaciales.

GSE0014E **La base de datos ya se ha habilitado para operaciones espaciales.**

Explicación: La base de datos ya se ha habilitado para operaciones espaciales.

Respuesta del usuario: Compruebe que la base de datos se ha habilitado del modo que esperaba. Si es necesario, inhabilite la base de datos.

GSE0498E **Se ha producido el siguiente error: "<mensaje-error>".**

GSE0499W **Spatial Extender ha emitido el siguiente aviso: "<mensaje-aviso>".**

GSE0500E **La modalidad de operación que ha especificado ("<modalidad-operación>") no es válida.**

Explicación: La modalidad especificada no recibe soporte de la operación que ha solicitado.

Respuesta del usuario: Consulte la documentación para ver qué modalidades reciben soporte de la operación.

GSE1001E **Spatial Extender no ha podido registrar una capa de vista denominada "<nombre-esquema.nombre-vista.nombre-columna>" que se basa en la columna espacial "<nombre-esquema.nombre-tabla.nombre-columna>".**

Explicación: La columna espacial que ha especificado ("<nombre-esquema.nombre-tabla.nombre-columna>") no se ha registrado como una capa de tabla.

Respuesta del usuario: Registre la columna "<nombre-esquema.nombre-tabla.nombre-columna>" como una capa de tabla.

GSE1002E Spatial Extender no ha podido registrar una capa de vista denominada "<nombre-esquema.nombre-vista.nombre-columna>" que se basa en la tabla "<nombre-esquema.nombre.tabla>".

Explicación: La tabla que ha especificado ("<nombre-esquema.nombre-tabla>") no es subyacente a la vista "<nombre-esquema.nombre-vista.nombre-columna>" ni directa ni indirectamente.

Respuesta del usuario: Busque cuál es la tabla base para la vista "<nombre-esquema.nombre-vista.nombre-columna>" y especifique dicha tabla.

GSE1003E Spatial Extender no ha podido acceder a la columna denominada "<nombre-columna>" de una tabla o vista denominada "<nombre-esquema.nombre-objeto>".

Explicación: La tabla o vista "<nombre-esquema.nombre-objeto>" no tiene ninguna columna denominada "<nombre-columna>".

Respuesta del usuario: Compruebe la definición de la tabla o vista "<nombre-esquema.nombre-objeto>" para ver el nombre adecuado de la columna que desea.

GSE1004E Spatial Extender no ha podido registrar la columna espacial calificada al completo "<nombre-esquema.nombre-tabla.nombre-columna>" como una capa de tabla.

Explicación: La columna "<nombre-esquema.nombre-tabla.nombre-columna>" no tiene un tipo de datos espaciales o no está asociada a una tabla base.

Respuesta del usuario: Defina un tipo de datos espaciales para la columna "<nombre-esquema.nombre-tabla.nombre-columna>", o asegúrese de que esta columna forma parte de una tabla base local.

GSE1005E El sistema de referencias espaciales ("<id-referencias-espaciales-capa-vista>") que ha especificado para una capa de vista difiere del sistema de referencias espaciales ("<id-referencias-espaciales-capa-tabla>") que se utiliza para la capa de tabla subyacente de esta capa.

Explicación: El sistema de referencias espaciales de una capa de vista debe coincidir con el sistema de referencias espaciales de la capa de tabla subyacente.

Respuesta del usuario: Especifique el sistema de referencias espaciales de la capa de tabla subyacente para la capa de vista.

GSE1006E Puesto que "<id-referencias-espaciales>" es un ID de sistema de referencias espaciales no válido, Spatial Extender no ha podido registrar la capa que ha solicitado.

Explicación: El sistema de referencias espaciales que ha especificado ("<id-referencias-espaciales>") no se ha habilitado o registrado.

Respuesta del usuario: Habilite o registre el sistema de referencias espaciales. Luego vuelva a someter la solicitud para registrar la capa.

GSE1007E Es posible que se haya producido un error de SQL (SQLSTATE "<estadosql>") cuando Spatial Extender ha intentado, sin éxito, añadir una columna espacial ("<nombre-columna>") a la tabla "<nombre-esquema.nombre-tabla>".

Respuesta del usuario: Busque el mensaje asociado a SQLSTATE "<estadosql>".

GSE1008E **DB2 Spatial Extender no ha podido registrar una capa de vista "<esquema-capa.nombre-capa.columna-capa>" porque el tipo de datos espaciales "<tipo-columna-capa>" de la capa de vista no coincide con el tipo de datos espaciales "<tipo-columna-geo>" de la capa de tabla subyacente "<esquema-geo.nombre-geo.columna-geo>".**

Explicación: El tipo de datos espaciales de una capa de vista "<esquema-capa.nombre-capa.columna-capa>" debe coincidir con el tipo de datos espaciales de la capa de tabla subyacente de la capa "<esquema-geo.nombre-geo.columna-geo>". La incoherencia entre estos dos tipos de datos causa ambigüedad cuando se procesan datos espaciales.

Respuesta del usuario: Asegúrese de que los tipos de datos espaciales de la capa de vista y de su capa de tabla subyacente coinciden.

GSE1020E **"<id-referencias-espaciales>" es un ID de sistema de referencias espaciales no válido.**

Explicación: No se ha habilitado un sistema de referencias espaciales con un identificador "<id-referencias-espaciales>".

Respuesta del usuario: Asegúrese de que la referencia espacial especificada se ha habilitado.

GSE1021E **Spatial Extender no ha podido habilitar el sistema de referencias espaciales "<id-referencias-espaciales>" porque el ID del sistema de coordenadas correspondiente "<id-coordenadas-espaciales>" no es válido.**

Explicación: No se ha definido ningún sistema de coordenadas con un identificador "<id-coordenadas-espaciales>" en el catálogo de Spatial Extender.

Respuesta del usuario: Compruebe el identificador del sistema de coordenadas "<id-coordenadas-espaciales>" consultando la vista DB2GSE.COORD_REF_SYS en el catálogo de Spatial Extender.

GSE1030E **Puesto que "<nombre-esquema.nombre-tabla>" no es una tabla base, Spatial Extender no ha podido habilitar un geocodificador para la misma.**

Explicación: El objeto que contiene los datos fuente que desea geocodificar debe ser una tabla base.

Respuesta del usuario: Asegúrese de que las columnas que contienen los datos fuente que desea codificar forman parte de una tabla base.

GSE1031E **Spatial Extender no ha podido habilitar el geocodificador "<id-geocodificador>" para que funcione automáticamente en modalidad de creación para la capa "<esquema-capa.nombre-capa.columna-capa>".**

Explicación: Las explicaciones posibles son las siguientes:

- El geocodificador ya está habilitado para actualizar la capa "<esquema-capa.nombre-capa.columna-capa>" automáticamente.
- El geocodificador se ha invalidado temporalmente para esta capa.
- No se ha definido ninguna columna para datos fuente para esta capa.

Respuesta del usuario: Si el geocodificador se ha invalidado temporalmente, habilítelo para que funcione automáticamente en modalidad de ″Recreación″.

GSE1032E Spatial Extender no ha podido habilitar el geocodificador "<id-geocodificador>" para que funcione automáticamente en modalidad de recreación para la capa "<esquema-capa.nombre-capa.columna-capa>".

Explicación: Las explicaciones posibles son las siguientes:

- El geocodificador ya está habilitado para actualizar la capa "<esquema-capa.nombre-capa.columna-capa>" automáticamente.
- El geocodificador no se ha invalidado previamente para esta capa.
- No se ha definido ninguna columna para datos fuente para esta capa.

Respuesta del usuario: Si el geocodificador se ha inhabilitado previamente en modalidad de eliminación, o si nunca se ha definido para esta capa, habilítelo para que funcione automáticamente en modalidad de ″Creación″.

GSE1033E Se ha producido un error de SQL cuando Spatial Extender intentaba añadir activadores a una tabla que contiene la columna para la capa "<esquema-capa.nombre-capa.columna-capa>" (SQLSTATE "<estadosql>").

Explicación: La finalidad de los activadores es mantener la integridad de los datos entre las columnas de atributo de las que procede la entrada del geocodificador y la columna espacial en la que va su salida. El error de SQL se ha producido cuando DB2 ha intentado crear estos activadores.

Respuesta del usuario: Busque el mensaje asociado a SQLSTATE "<estadosql>".

GSE1034E Spatial Extender no ha podido inhabilitar el geocodificador "<id-geocodificador>" en modalidad de eliminación para la capa "<esquema-capa.nombre-capa.columna-capa>".

Explicación: Las explicaciones posibles son las siguientes:

- El geocodificador nunca se ha habilitado para actualizar la capa "<esquema-capa.nombre-capa.columna-capa>" automáticamente.
- El geocodificador se ha inhabilitado en modalidad de eliminación.

Respuesta del usuario: Determine el estado en que estaba el geocodificador antes de que intentara inhabilitarlo. Por ejemplo, ¿estaba registrado?, ¿estaba habilitado? Luego decida si se tiene que inhabilitar en modalidad de eliminación. Por ejemplo, si nunca se ha habilitado, no hay ninguna necesidad de inhabilitarlo.

GSE1035E Spatial Extender no ha podido inhabilitar el geocodificador "<id-geocodificador>" en modalidad de invalidación para la capa "<esquema-capa.nombre-capa.columna-capa>".

Explicación: Las explicaciones posibles son las siguientes:

- El geocodificador nunca se ha habilitado para actualizar la capa "<esquema-capa.nombre-capa.columna-capa>" automáticamente.
- El geocodificador se ha inhabilitado en modalidad de invalidación o en modalidad de eliminación.

Respuesta del usuario: Determine el estado en que estaba el geocodificador antes de que intentara inhabilitarlo. Por ejemplo, ¿estaba registrado?, ¿estaba habilitado? Luego decida si se tiene que inhabilitar en modalidad de invalidación. Por ejemplo, si ya se había inhabilitado en modalidad de invalidación, no hay ninguna necesidad de inhabilitarlo por segunda vez en esta modalidad.

GSE1036E Se ha producido un error de SQL cuando Spatial Extender intentaba eliminar activadores desde una tabla que contiene la columna para la capa "<esquema-capa.nombre-capa.columna-capa>" (SQLSTATE "<estadosql>").

Explicación: Los activadores se crearon para mantener la integridad de los datos entre las columnas de atributo de las que procede la entrada del geocodificador y la columna espacial en la que va su salida. El error de SQL se ha producido cuando DB2 intentaba eliminar estos activadores.

Respuesta del usuario: Busque el mensaje asociado a SQLSTATE "<estadosql>".

GSE1037E Spatial Extender no ha podido geocodificar datos fuente para la capa de tabla "<esquema-capa.nombre-capa.columna-capa>", posiblemente porque se ha asignado un valor incorrecto "<número-de-atributos>" al argumento que especifica el número de columnas de atributo que deben suministrar datos fuente para esta capa.

Explicación: El número de columnas de atributo asociado a esta capa se ha especificado incorrectamente, o bien el nombre de una o más de estas columnas se ha especificado incorrectamente.

Respuesta del usuario: Asegúrese de que esta capa está registrada con el número y nombres correctos de columnas de atributo asociadas o bien compruebe que los datos de entrada y salida del geocodificador son correctos.

GSE1038E Se ha producido un error de SQL cuando Spatial Extender intentaba geocodificar datos fuente para la capa de tabla "<esquema-capa.nombre-capa.columna-capa>" en modalidad de proceso por lotes (SQLSTATE "<estadosql>").

Respuesta del usuario:

- Busque el mensaje asociado a SQLSTATE "<estadosql>".
- Asegúrese de que el contenido y el argumento primaryUDF de esta capa se han definido correctamente.

GSE1050E El tamaño de cuadrícula que ha especificado ("<tamaño-cuadrícula>") no es válido para el primer nivel de cuadrícula.

Explicación: Ha especificado cero o un número negativo como el tamaño de cuadrícula correspondiente al primer nivel de cuadrícula.

Respuesta del usuario: Especifique un número positivo como el tamaño de cuadrícula.

GSE1051E El tamaño de cuadrícula que ha especificado ("<tamaño-cuadrícula>") no es válido para los niveles segundo y tercero de cuadrícula.

Explicación: Ha especificado un número negativo como el tamaño de cuadrícula correspondiente al segundo o tercer nivel de cuadrícula.

Respuesta del usuario: Especifique cero o un número positivo como el tamaño de cuadrícula.

GSE1052E Se ha producido un error de SQL cuando Spatial Extender intentaba crear el índice espacial "<esquema-índice.columna-índice>" para una capa de tabla "<esquema-capa.nombre-capa.columna-capa>" (SQLSTATE "<estadosql>").

Respuesta del usuario:

- Asegúrese de que el índice espacial se ha especificado correctamente y que la columna espacial no tiene ningún índice asociado.
- Busque el mensaje asociado a SQLSTATE "<estadosql>".

GSE1500I El registro fuente "<número-registro>" se ha geocodificado satisfactoriamente.

Explicación: Un registro que contiene datos de atributo se ha geocodificado satisfactoriamente.

GSE1501W El registro fuente "<número-registro>" no se ha geocodificado.

Explicación: El nivel de precisión era demasiado alto.

Respuesta del usuario: Geocodifique con un nivel de precisión más bajo.

GSE1502W No se ha encontrado el registro fuente "<número-registro>".

Respuesta del usuario: Determine si el registro existe en la base de datos.

GSE2001E El archivo de transferencia especificado ("<nombrearchivo>") no es válido.

Respuesta del usuario: Compruebe que el archivo especificado es un archivo de transferencia SDE y que la vía de acceso se ha especificado correctamente.

GSE2002E La cláusula SQL WHERE suministrada ("<cláusula-SQL-where>") no es válida.

Respuesta del usuario: Compruebe la cláusula WHERE para ver si la sintaxis SQL es correcta, si contiene errores de escritura o si contiene nombres de columna no válidos.

GSE2003E El valor de forma suministrado no es válido.

Respuesta del usuario: Asegúrese de que la forma suministrada coincide con el tipo especificado de la columna espacial. Si los tipos coinciden, o son compatibles, significa que el formato de geometría no es válido. Compruebe si hay polígonos que se solapan, arcos de un solo punto, etc.

GSE2004E El esquema de archivo de transferencia es incompatible con el esquema de la capa especificada.

Respuesta del usuario: Compruebe que los nombres del esquema y de la capa se han especificado correctamente. Si los esquemas no coinciden, cargue los datos como una tabla nueva y resuelva las diferencias de esquema.

GSE2005E El tipo de geometría del archivo de transferencia es incompatible con el tipo de geometría de la capa especificada.

Respuesta del usuario: Compruebe que los nombres del esquema y de la capa se han especificado correctamente.

GSE2006E Se ha producido un error de E/S correspondiente a un archivo denominado "<nombrearchivo>".

Respuesta del usuario: Compruebe que el archivo existe, que tiene el acceso adecuado al archivo y que este no está siendo utilizado por otro usuario.

GSE2007E Se ha producido un error de conversión de atributo.

Respuesta del usuario: Compruebe que todos los tipos de atributo de la tabla reciben soporte - por ejemplo, los datos BLOB no reciben soporte en archivos de forma. Compruebe también si hay valores de datos fuera del rango válido o valores de datos no válidos, como fechas incorrectas.

GSE2008E La función de importación/exportación se ha quedado sin memoria.

Respuesta del usuario: Compruebe que dispone de la memoria adecuada.

Capítulo 11. Vistas de catálogo

Las vistas de catálogo de DB2 Spatial Extender contienen metadatos sobre:

- Sistemas de coordenadas que puede utilizar. Para obtener información como textos de anotación e identificadores de estos sistemas, consulte el tema "DB2GSE.COORD_REF_SYS".
- Columnas espaciales que se han registrado como capas. Para obtener información como los nombres, los tipos de datos y los sistemas de referencias espaciales asociados a estas columnas, consulte el tema "DB2GSE.GEOMETRY_COLUMNS" en la página 118.
- Geocodificadores que puede utilizar. Para obtener información como los identificadores de estos geocodificadores y las regiones que contienen las ubicaciones que procesan los geocodificadores, consulte el tema "DB2GSE.SPATIAL_GEOCODER" en la página 119.
- Sistemas de referencias espaciales que puede utilizar. Para obtener información como sus identificadores y descripciones, consulte el tema "DB2GSE.SPATIAL_REF_SYS" en la página 120.

DB2GSE.COORD_REF_SYS

Cuando habilita una base de datos para operaciones espaciales, DB2 Spatial Extender registra los sistemas de coordenadas que puede utilizar en una tabla de catálogo. Las columnas seleccionadas de esta tabla forman la vista de catálogo DB2GSE.COORD_REF_SYS, que se describe en la Tabla 34.

Tabla 34. Columnas de la vista de catálogo DB2GSE.COORD_REF_SYS

Nombre	Tipo de datos	¿Posibilidad de nulos?	Contenido
SCID	INTEGER	No	Identificador numérico exclusivo para este sistema de coordenadas.
SC_NAME	VARCHAR(64)	No	Nombre de este sistema de coordenadas.
AUTH_NAME	VARCHAR(256)	Sí	Sistema de coordenadas con el que cumple la organización que ha compilado este sistema de coordenadas; por ejemplo, European Petroleum Survey Group (EPSG).
AUTH_SRID	INTEGER	Sí	Un identificador numérico asignado a este sistema de coordenadas por la organización especificada en la columna AUTH_NAME.
DESC	VARCHAR(256)	Sí	Descripción de este sistema de coordenadas.

Tabla 34. Columnas de la vista de catálogo DB2GSE.COORD_REF_SYS (continuación)

Nombre	Tipo de datos	¿Posibilidad de nulos?	Contenido
SRTEXT	VARCHAR(2048)	No	Texto de anotación para este sistema de coordenadas.

DB2GSE.GEOMETRY_COLUMNS

Cuando crea una capa, DB2 Spatial Extender la registra grabando su identificador e información relacionada con la misma en la tabla de catálogo. Las columnas seleccionadas de esta tabla forman la vista de catálogo DB2GSE.GEOMETRY_COLUMNS, que se describe en la Tabla 35.

Tabla 35. Columnas de la vista de catálogo DB2GSE.GEOMETRY_COLUMNS

Nombre	Tipo de datos	¿Posibilidad de nulos?	Contenido
LAYER_CATALOG	VARCHAR(30)	Sí	Nombre calificado al completo de esta capa.
LAYER_SCHEMA	VARCHAR(30)	No	Esquema de la tabla o vista que contiene la columna que se ha registrado como esta capa.
LAYER_NAME	VARCHAR(128)	No	Nombre de la tabla o vista que contiene la columna que se ha registrado como esta capa.
LAYER_COLUMN	VARCHAR(30)	No	Nombre de la columna que se ha registrado como esta capa.
GEOMETRY_TYPE	INTEGER	No	Tipo de datos de la columna que se ha registrado como esta capa.
SRID	INTEGER	No	Identificador del sistema de referencias espaciales utilizado para los valores de la columna que se ha registrado como esta capa.
STORAGE_TYPE	INTEGER	Sí	Información sobre cómo DB2 almacena los valores de la columna que se ha registrado como esta capa. Por ejemplo, los datos de STORAGE_TYPE pueden indicar que los valores se almacenan como objetos grandes (LOB) o como instancias de tipos de datos abstractos (ADT).

DB2GSE.SPATIAL_GEOCODER

Los geocodificadores disponibles se registran en una tabla de catálogo. Las columnas seleccionadas de esta tabla forman la vista de catálogo DB2GSE.SPATIAL_GEOCODER, que se describe en la Tabla 36.

Tabla 36. Columnas de la vista de catálogo DB2GSE.SPATIAL_GEOCODER

Nombre	Tipo de datos	¿Posibilidad de nulos?	Contenido
GCID	INTEGER	No	Identificador numérico de este geocodificador.
GC_NAME	VARCHAR(64)	No	Breve descripción de este geocodificador.
VENDOR_NAME	VARCHAR(128)	No	Nombre del proveedor que ha suministrado este geocodificador.
PRIMARY_UDF	VARCHAR(256)	No	Nombre calificado al completo de este geocodificador.
PRECISION_LEVEL	INTEGER	No	El grado en que los datos fuente deben coincidir con los datos de referencia correspondientes para que el geocodificador los procese satisfactoriamente.
VENDOR_SPECIFIC	VARCHAR(256)	Sí	Nombre y vía de acceso de un archivo que puede utilizar un proveedor para definir parámetros especiales a los que da soporte este geocodificador.
GEO_AREA	VARCHAR(256)	Sí	Área geográfica que contiene las ubicaciones que se tienen que geocodificar.
DESCRIPTION	VARCHAR(256)	Sí	Observaciones suministradas por el proveedor.

DB2GSE.SPATIAL_REF_SYS

Cuando crea un sistema de referencias espaciales, DB2 Spatial Extender lo registra grabando su identificador e información relacionada con el mismo en una tabla de catálogo. Las columnas seleccionadas de esta tabla forman la vista de catálogo DB2GSE.SPATIAL_REF_SYS, que se describe en la Tabla 37.

Tabla 37. Columnas de la vista de catálogo DB2GSE.SPATIAL_REF_SYS

Nombre	Tipo de datos	¿Posibilidad de nulos?	Contenido
SRID	INTEGER	No	Identificador definido por el usuario de este sistema de referencias espaciales.
SR_NAME	VARCHAR(64)	No	Nombre de este sistema de referencias espaciales.
SCID	INTEGER	No	Identificador numérico del sistema de coordenadas subyacente a este sistema de referencias espaciales.
SC_NAME	VARCHAR(64)	No	Nombre del sistema de coordenadas subyacente a este sistema de referencias espaciales.
AUTH_NAME	VARCHAR(256)	Sí	Nombre de la organización que define los estándares correspondientes a este sistema de referencias espaciales.
AUTH_SRID	INTEGER	Sí	El identificador que la organización especificada en la columna AUTH_NAME asigna a este sistema de referencias espaciales.
SRTEXT	VARCHAR(2048)	No	Texto de anotación correspondiente a este sistema de referencias espaciales.

Capítulo 12. Índices espaciales

Puesto que las columnas espaciales contienen datos geográficos de dos dimensiones, las aplicaciones que efectúan consultas en dichas columnas necesitan una estrategia de índice que identifique rápidamente todas las geometrías contenidas en una determinada extensión. Por este motivo, DB2 Spatial Extender ofrece el índice espacial de tres niveles basado en una cuadrícula.

Este capítulo describe este tipo de índice y contiene directrices para utilizarlo. Los temas de esta sección son:

Un fragmento de programa de ejemplo

Utilizaremos el siguiente ejemplo para ver cómo se crea un índice y se utiliza en SQL. Consulte el manual *SQL Reference* para obtener más información sobre los mandatos CREATE INDEX y CREATE INDEX EXTENSION. Tenga en cuenta que, una vez creado el índice, puede emitir sentencias DDL y DML estándares que utilicen predicados y funciones espaciales.

```
create table customers (cid int, addr varchar(40), ..., loc db2gse.ST_Point)
create table stores (sid int, addr varchar(40), ..., loc db2gse.ST_Point,
 zone db2gse.ST_Polygon)

create index customersx1 on customers(loc) extend using spatial_index(10e0,
 100e0, 1000e0)
create index storesx1 on stores(loc) extend using spatial_index(10e0, 100e0,
 1000e0)
create index storesx2 on stores(zone) extend using spatial_index(10e0, 100e0,
 1000e0)

insert into customers (cid, addr, loc) values
(:cid, :addr, sdeFromBinary(:loc))
insert into customers (cid, addr, loc) values
(:cid, :addr, geocode(:addr))
insert into stores (sid, addr, loc) values
(:sid, :addr, sdeFromBinary(:loc))

update stores set zone = db2gse.ST_Buffer (loc, 2)

select cid, loc from customers
```

```
 where db2gse.ST_Within(loc, :polygon) = 1

select cid, loc from customers
 where db2gse.ST_Within(loc, :circle1) = 1 OR
       db2gse.ST_Within(loc, :circle2) = 1

select c.cid, loc from customers c, stores s
 where db2gse.ST_Contains(s.zone, c.loc) = 1 selectivity 0.01

select avg(c.income) from customers c
 where not exist (select * from stores s
                  where db2gse.ST_Distance(c.loc, s.loc) < 10)
```

Índices de árbol B

La tecnología de indexación espacial se basa en el índice de árbol B jerárquico tradicional, pero es significativamente diferente. El índice espacial utiliza *indexación de cuadrícula* que está diseñada para indexar columnas espaciales de dos dimensiones. El índice de árbol B sólo puede manejar datos de una dimensión y no se puede utilizar con información GIS. Esta sección describe cómo se estructura y se utiliza un índice de árbol B.

El nivel superior de un índice de árbol B, denominado nodo raíz, contiene una clave para cada nodo del siguiente nivel. El valor de cada clave es el valor de clave mayor existente para el nodo correspondiente del siguiente nivel. En función del número de valores de la tabla base, es posible que se necesiten varios nodos intermedios. Estos nodos forman un puente entre el nodo raíz y los nodos hoja que alojan los ID reales de filas de la tabla base.

El gestor de la base de datos busca en un índice de árbol B empezando por el nodo raíz. Luego continúa por los nodos intermedios hasta alcanzar el nodo hoja con el ID de fila de la tabla base.

El índice de árbol B no se puede aplicar a una columna espacial porque la característica de dos dimensiones de la columna espacial necesita la estructura de un índice espacial. Por el mismo motivo, no puede aplicar un índice espacial a una columna que no sea espacial. Además, un índice espacial no se puede aplicar a una columna compuesta de ningún tipo.

Formas de crear un índice espacial

Hay varias formas de crear un índice espacial:

- Definiendo uno en la ventana Crear índice espacial. Para obtener instrucciones, consulte el “Capítulo 6. Creación de índices espaciales” en la página 53.

- Invocando el procedimiento almacenado db2gse.gse_enable_idx en un programa de aplicación. Para obtener información sobre este procedimiento almacenado, consulte el “Capítulo 9. Procedimientos almacenados” en la página 69.
- Emitiendo el mandato **db2 create index** con la función **spatial_index** en la cláusula USING, Por ejemplo:

```
create index storesx1 on customers (loc) using spatial_index(10e0,
100e0, 1000e0)
```

La naturaleza de los datos espaciales requiere que el diseñador de la base de datos comprenda su distribución relativa de tamaños. El diseñador debe determinar el tamaño óptimo y el número de niveles de cuadrícula con los que crear el índice espacial.

Los niveles de cuadrícula, <nivel de cuadrícula 1>, <nivel de cuadrícula 2> y <nivel de cuadrícula 3>, se especifican aumentando el tamaño de celda. Por lo tanto, el segundo nivel debe tener un tamaño de celda superior al del primero, y el tercero uno superior al del segundo. El primer nivel de cuadrícula es obligatorio, pero puede inhabilitar el segundo y el tercer nivel con un valor cero de doble precisión (0.0e0).

Cómo se genera un índice espacial

Un índice espacial se genera utilizando *envolturas*. La envoltura es una geometría y representa la extensión X e Y mínima y máxima de una geometría. Para la mayoría de las geometrías, la envoltura es un recuadro, pero para series lineales horizontales y verticales, la envoltura es una serie lineal de dos puntos. Para puntos, la envoltura es un punto. Para obtener más información sobre envolturas, consulte el tema “Envoltura” en la página 136.

El índice espacial se construye en una columna espacial creando una o más entradas correspondientes a las intersecciones de cada envoltura de geometría con la cuadrícula. Una intersección se registra como el ID interno de la geometría y las coordenadas X e Y mínimas de la celda de la cuadrícula que forma la intersección. Por ejemplo, el polígono de la Figura 7 en la página 124 forma intersección con la cuadrícula en las coordenadas (20,30), (30,30), (40,30), (20,40), (30,40), (40,40), (20,50), (30,50) y (40,50). Consulte la Tabla 38 en la página 125 para ver las coordenadas X e Y mínimas para todas las geometrías de la Figura 7 en la página 124.

Si hay varios niveles de cuadrícula, DB2 Spatial Extender intenta utilizar el nivel de cuadrícula más bajo posible. Cuando una geometría forma intersección con cuatro o más celdas de cuadrícula a un determinado nivel, se promociona al nivel superior siguiente. Por lo tanto, para un índice espacial que tiene los tres niveles de cuadrícula de 10.0e0, 100.0e0 y 1000.0e0, DB2 Spatial Extender formará primero intersección entre cada geometría con la

cuadrícula de nivel 10.0e0. Si una geometría forma intersección con cuatro o más celdas de cuadrícula 10.0e0, se promociona y formará intersección con la cuadrícula de nivel 100.0e0. Si hay cuatro o más intersecciones al nivel 100.0e0, la geometría se promociona al nivel 1000.0e0. En el nivel 1000.0e0, las intersecciones se deben entrar en el índice espacial porque es el nivel más alto posible.

La Figura 7 ilustra cómo cuatro tipos diferentes de geometrías forman intersección con una cuadrícula 10.0e. Las 23 intersecciones correspondientes a las cuatro geometrías se registran en el índice espacial.

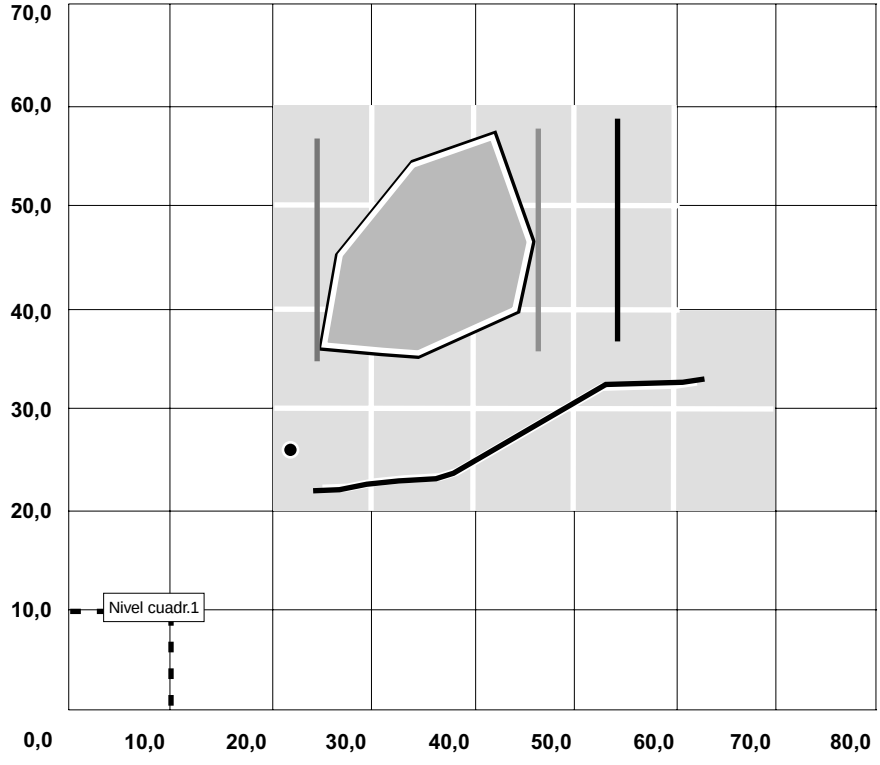


Figura 7. Aplicación a un nivel de cuadrícula 10.0e0

La Tabla 38 en la página 125 contiene las geometrías y sus correspondientes intersecciones de cuadrícula. Las envolturas de cuatro tipos de geometrías diferentes forman intersección con la cuadrícula 10.0e. La coordenada X e Y mínima de cada celda de cuadrícula que forma intersección se entra en el índice espacial.

Tabla 38. Las entradas de celda de cuadrícula 10.0e0 correspondientes a las geometrías de ejemplo

Geometría	Cuadrícula X	Cuadrícula Y
Polígono	20,0	30,0
Polígono	30,0	30,0
Polígono	40,0	30,0
Polígono	20,0	40,0
Polígono	30,0	40,0
Polígono	40,0	40,0
Polígono	20,0	50,0
Polígono	30,0	50,0
Polígono	40,0	50,0
Serie lineal vertical	50,0	30,0
Serie lineal vertical	50,0	40,0
Serie lineal vertical	50,0	50,0
Punto	20,0	20,0
Serie lineal horizontal	20,0	20,0
Serie lineal horizontal	30,0	20,0
Serie lineal horizontal	40,0	20,0
Serie lineal horizontal	50,0	20,0
Serie lineal horizontal	60,0	20,0
Serie lineal horizontal	20,0	30,0
Serie lineal horizontal	30,0	30,0
Serie lineal horizontal	40,0	30,0
Serie lineal horizontal	50,0	30,0
Serie lineal horizontal	60,0	30,0

La Figura 8 en la página 126 muestra cómo el número de intersecciones se reduce significativamente a ocho al añadir los niveles de cuadrícula 30.0e0 y 60.0e0. En este caso, el polígono identificado como geometría 1 se promociona al nivel de cuadrícula 30.0e0 y la serie lineal identificada como geometría 4 se promociona al nivel de cuadrícula 60.0e0. En lugar de las nueve y diez intersecciones que tenían las geometrías en el nivel 10.0e0, sólo tienen dos tras la promoción.

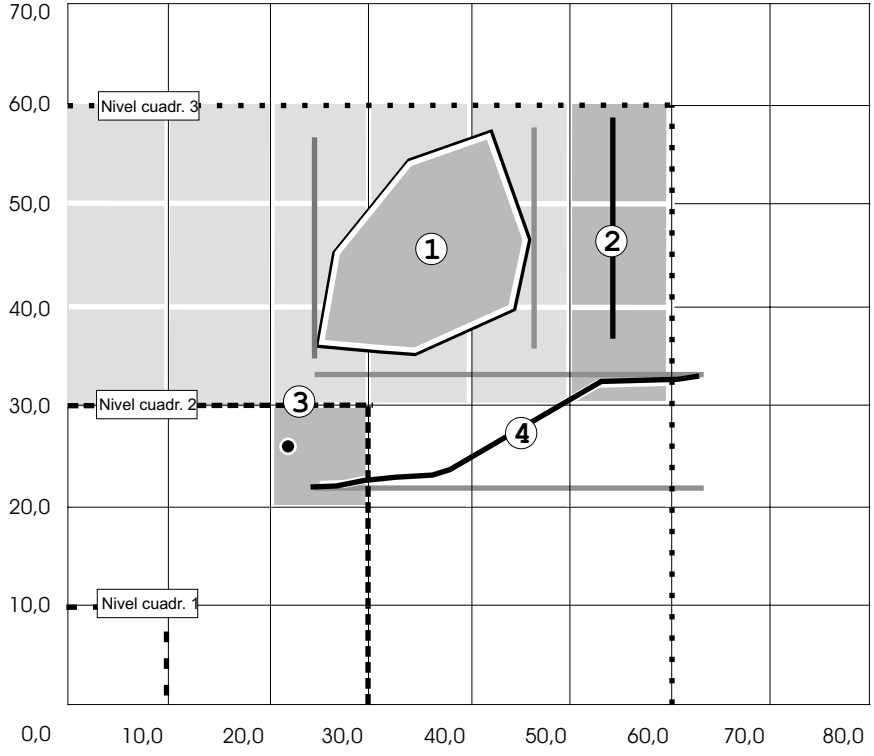


Figura 8. Efecto de añadir los niveles de cuadrícula 30.0e0 y 60.0e0. La envoltura del polígono identificado como geometría 1 forma intersección con nueve celdas de cuadrícula. La envoltura de la serie lineal vertical identificada como geometría 2 forma intersección con tres celdas de cuadrícula. La envoltura del punto identificado como geometría 3 forma intersección con una sola celda de cuadrícula. La envoltura de la serie lineal identificada como geometría 4 forma intersección con diez celdas de cuadrícula.

DB2 Spatial Extender toma los parámetros de nivel de cuadrícula especificados en la sentencia CREATE INDEX y comprueba cada objeto espacial para determinar las coordenadas y el número de bloques de cuadrícula en los que se encuentra cada objeto. En la Figura 8, los niveles de cuadrícula 10.0e0, 30.0e0 y 60.0e0 se muestran con pesos lineales crecientes y diferentes tonos de gris. Las intersecciones de las celdas de las envolturas de la serie lineal vertical y del punto se entran en el índice al nivel de cuadrícula 10.0e0, porque ambas generan menos de cuatro intersecciones. El polígono forma intersección con nueve celdas de la cuadrícula 10.0e0, por lo que se promociona al nivel de cuadrícula 30.0e0. A este nivel, el polígono forma intersección con dos celdas de cuadrícula, que se entran en el índice. La serie lineal identificada como geometría 4 forma intersección con diez celdas de la cuadrícula 10.0e0, por lo que se promociona al nivel de cuadrícula 30.0e0. Ya a este nivel, forma intersección con seis celdas de cuadrícula, por lo que se vuelve a promocionar al nivel de cuadrícula 60.0e0, donde genera dos intersecciones. Las intersecciones de la cuadrícula 60.0e0 de la serie lineal se

entran en el índice. Si la serie lineal hubiera generado cuatro o más intersecciones a este nivel, se entraría igualmente en el índice porque es el nivel superior al que se puede promocionar una geometría.

Tabla 39. Las intersecciones de las geometrías en el índice de tres niveles

Geometría	Cuadrícula X	Cuadrícula Y
Las intersecciones entre la serie lineal vertical y el punto en el nivel 1 (tamaño de cuadrícula 10.0e0)		
2	50,0	30,0
2	50,0	40,0
2	50,0	50,0
3	20,0	20,0
Las intersecciones del polígono en el nivel 2 (tamaño de cuadrícula 30.0e0)		
1	0,0	30,0
1	30,0	30,0
Las intersecciones de la serie lineal en el nivel 3 (tamaño de cuadrícula 60.0e0)		
4	0,0	0,0
4	60,0	0,0

DB2 Spatial Extender realmente no crea ninguna estructura de cuadrículas de polígono de ningún tipo. DB2 Spatial Extender manifiesta cada nivel de cuadrícula de forma paramétrica definiendo el origen en el desplazamiento X,Y del sistema de referencias espaciales de las columnas. Luego amplía la cuadrícula a un espacio de coordenadas positivas. Utilizando una cuadrícula paramétrica, DB2 Spatial Extender genera las intersecciones de forma matemática.

Directrices sobre la utilización de un índice espacial

DB2 Spatial Extender trabaja con un índice espacial para mejorar el rendimiento de una consulta espacial. Supongamos que tenemos la consulta espacial más básica y que se utiliza con más frecuencia, la consulta de recuadro. Esta consulta solicita a DB2 Spatial Extender que devuelva todas las geometrías que se encuentran total o parcialmente dentro de un recuadro definido por el usuario. Si no existe un índice, DB2 Spatial Extender debe comparar todas las geometrías con el recuadro. Sin embargo, con un índice DB2 Spatial Extender puede localizar todas las entradas de índice que tienen una coordenada inferior izquierda superior o igual a la del recuadro y una coordenada superior derecha inferior o igual a la del recuadro. Puesto que el sistema de coordenadas clasifica el índice, DB2 Spatial Extender puede obtener rápidamente una lista de geometrías candidatas. El proceso que se acaba de describir recibe el nombre de *primera pasada*.

Una *segunda pasada* determina si la envoltura de cada candidato forma intersección con el recuadro. Una geometría calificada para la primera pasada porque la envoltura de sus celdas de cuadrícula forma intersección con el recuadro puede tener una envoltura que no lo haga.

Una *tercera pasada* compara las coordenadas reales del candidato con el recuadro para determinar si parte de la geometría se encuentra realmente dentro del recuadro. El último y más complejo proceso de comparación opera sobre una lista de candidatos compuesta por un subconjunto de la población total, que se reduce significativamente tras las dos primeras pasadas.

Todas las consultas espaciales llevan a cabo las tres pasadas excepto la función EnvelopesIntersect. Esta sólo lleva a cabo las dos primeras pasadas. La función EnvelopesIntersect se ha diseñado para operaciones de visualización que suelen utilizar sus propias rutinas de recorte incorporadas y no necesitan la granularidad de la tercera pasada.

Selección del tamaño de celda de cuadrícula

La forma irregular de las envolturas de las geometrías complica la selección del tamaño de celda de la cuadrícula. Debido a esta irregularidad, algunas envolturas de geometrías forman intersección con varias cuadrículas, mientras que otras caben dentro de una sola celda de cuadrícula. A la inversa, en función de la distribución espacial de los datos, algunas celdas de cuadrícula forman intersección con varias envolturas de geometrías.

Para que un índice espacial funcione bien, resulta esencial seleccionar el número y el tamaño correcto de cuadrículas. Supongamos que tenemos una columna espacial que contiene geometrías de tamaño uniforme. En este caso, un solo nivel de cuadrícula será suficiente. Comience por un tamaño de celda de cuadrícula que comprenda la envoltura media de las geometrías. Mientras prueba su aplicación, puede descubrir que al aumentar el tamaño de celda de cuadrícula mejora el rendimiento de sus consultas. Esto se debe a que cada celda de cuadrícula contiene más geometrías y la primera pasada puede eliminar las geometrías no calificadas con mayor rapidez. Sin embargo, descubrirá que a medida que continúa aumentando el tamaño de celda, el rendimiento se va deteriorando. Esto se debe probablemente a que la segunda pasada tiene que enfrentarse a más candidatos.

Selección del número de niveles

Si los objetos que desea indexar tienen aproximadamente el mismo tamaño relativo, puede utilizar un solo nivel de cuadrícula. Aunque esto es cierto, no todas las columnas contendrán geometrías del mismo tamaño relativo. Normalmente, las geometrías de columnas espaciales se pueden agrupar en varios intervalos de tamaños. Por ejemplo, supongamos que tenemos una red de carreteras en la que las geometrías se dividen en calles, carreteras principales y autopistas. Todas las calles tienen aproximadamente la misma

longitud y se pueden agrupar en un intervalo de tamaños. Esto también se aplica a las carreteras principales y a las autopistas. Por lo tanto, las calles, que representan un intervalo de tamaños, se pueden agrupar en el primer nivel de cuadrícula, las redes de carreteras en el segundo y las autopistas en el tercer nivel de cuadrícula. Otro ejemplo incluye una columna de parcelas que contiene agrupaciones de pequeñas parcelas urbanas rodeadas de parcelas rurales de mayor tamaño. En este caso, hay dos intervalos de tamaños y dos niveles de cuadrícula, uno para las pequeñas parcelas urbanas y otro para las parcelas rurales de mayor tamaño. Estas situaciones son muy comunes y requieren el uso de una cuadrícula de varios niveles.

Para seleccionar el tamaño de celda de cada nivel de cuadrícula, seleccione tamaños de celda de cuadrícula ligeramente superiores que cada intervalo de tamaños. Compruebe el índice realizando consultas sobre la columna espacial.

Cada nivel adicional necesita otra exploración de índice. Vaya aumentando y disminuyendo ligeramente los tamaños de cuadrícula si consigue con ello una mejora apreciable en el rendimiento.

Capítulo 13. Geometrías y funciones espaciales asociadas

Este capítulo trata sobre unidades de información, denominadas *geometrías*, que constan de coordenadas y simbolizan funciones geográficas. También contiene una introducción a las funciones espaciales que toman geometrías como entrada y devuelven resultados que le ayudan a analizar funciones geográficas y a mover datos espaciales entre sistemas de información geográfica. Los temas de esta sección son:

- La naturaleza de las geometrías
- Propiedades de las geometrías: funciones que devuelven información relacionada con estas propiedades
- Geometrías con las que se pueden crear instancias; funciones que se pueden llevar a cabo sobre las mismas
- Funciones que:
  - Muestran relaciones y comparaciones entre funciones geográficas
  - Generan geometrías
  - Convierten valores geométricos en formatos que se pueden importar y exportar

Acerca de las geometrías

Oxford American Dictionary define *geometría* como "la rama de las matemáticas que maneja las propiedades y las relaciones entre líneas, ángulos, superficies y formas tridimensionales." El 11 de agosto de 1997, el denominado Open GIS Consortium Inc. (OGC), en su publicación *Open GIS Features for ODBC (SQL) Implementation Specification*, acuñó otra definición para el término. Se seleccionó la palabra *geometría* para indicar funciones geométricas que, durante el siglo pasado o incluso antes, utilizaron los cartógrafos para generar el mapa del mundo. Una definición muy abstracta de este nuevo significado podría ser "un punto o agregación de puntos que simbolizan una función en la tierra".

En DB2 Spatial Extender, una definición *operativa* de geometría sería "un modelo de una función geográfica". El modelo se puede expresar mediante coordenadas de la función y, en algunos casos, mediante un símbolo visual. El modelo contiene información; por ejemplo, las coordenadas identifican la posición de la función con respecto a puntos de referencia fijos y el símbolo indica su forma. Además, el modelo se puede utilizar para generar información; por ejemplo, la función ST_Overlaps puede tomar las coordenadas de dos regiones próximas como entrada y devolver información sobre si las regiones se solapan o no.

Las coordenadas de una función simbolizada por una geometría reciben el nombre de *propiedades* de la geometría. Otros tipos de geometrías tienen otras propiedades; por ejemplo:

- Un *interior* representa el contenido de la función simbolizada por la geometría.
- Un *exterior* representa el espacio que rodea la función.
- Un *límite* representa la demarcación donde termina el contenido y comienza el espacio que le rodea.

Estas y otras propiedades se describen en el tema “Propiedades de geometrías y funciones asociadas” en la página 133.

Las geometrías que reciben soporte de DB2 Spatial Extender forman una jerarquía, que se muestra en la Figura 9. Se pueden crear instancias de seis miembros de la jerarquía; se pueden expresar como símbolos visuales, que también se muestran en la figura.

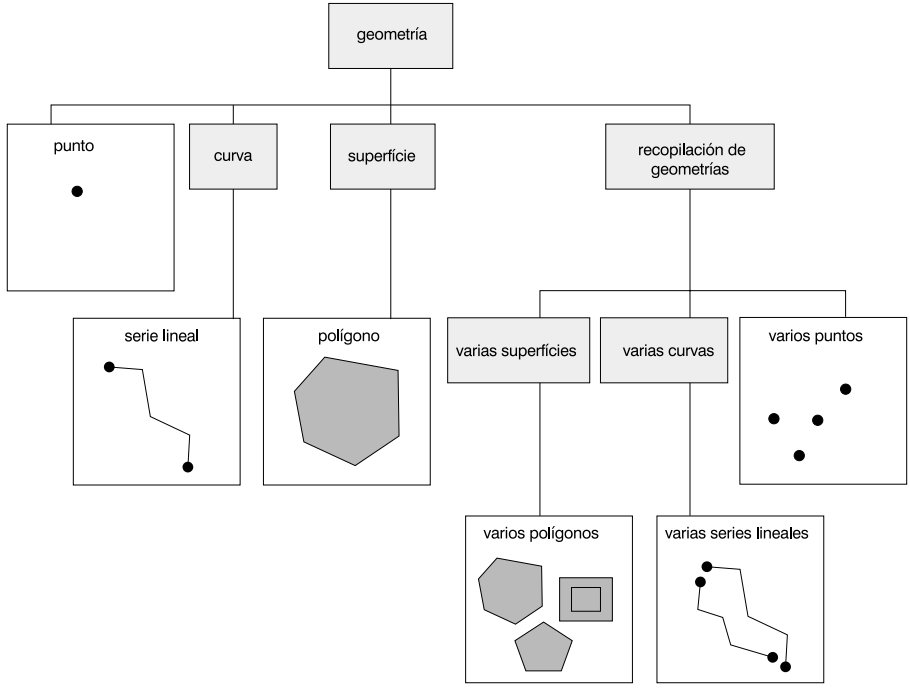


Figura 9. Jerarquía de geometrías que reciben soporte de DB2 Spatial Extender. Las geometrías con las que se pueden crear instancias se pueden expresar como símbolos visuales. Estos símbolos se muestran bajo los nombres de estas geometrías.

Tal como indica la Figura 9 en la página 132, una superclase denominada *geometría* es la raíz de la jerarquía. Las subclases se dividen en dos categorías: las subclases de geometría base y las subclases de recopilación homogénea. Las geometrías base incluyen:

- *Puntos*, que simbolizan funciones distinguidas que se perciben como funciones que ocupan el lugar geométrico donde una línea de coordenadas este-oeste (como un paralelo) forma intersección con una línea de coordenadas norte-sur (como un meridiano). Por ejemplo, supongamos que la notación en un mapa a gran escala muestra que cada ciudad del mapa se encuentra en la intersección entre un paralelo y un meridiano. En esta escala, cada ciudad se podría simbolizar mediante un punto.
- *Series lineales*, que simbolizan funciones geográficas lineales (por ejemplo, calles, canales y conductos).
- *Polígonos*, que simbolizan funciones geográficas de varios lados (por ejemplo, distritos de prestaciones sociales, bosques o hábitats de vida salvaje).

Las recopilaciones homogéneas incluyen:

- *Varios puntos*, que simbolizan funciones que constan de varias partes, cuyos componentes se encuentran cada uno en la intersección de una línea de coordenadas este-oeste y una línea de coordenadas norte-sur (por ejemplo, una cadena de islas cuyos miembros se encuentran cada uno de ellos situado en la intersección entre un paralelo y un meridiano).
- *Varias series lineales*, que simbolizan funciones de varias partes formadas por componentes o unidades lineales (por ejemplo, sistemas de ríos y sistemas de autopistas).
- *Varios polígonos*, que simbolizan funciones de varias partes formadas por componentes o unidades de varios lados (por ejemplo, el grupo de granjas de una determinada región o un sistema de lagos).

Tal como indican sus nombres, las recopilaciones homogéneas son grupos de geometrías base. Además de compartir las propiedades de las geometrías base, las recopilaciones homogéneas tienen también sus propias propiedades.

Los tipos de datos espaciales que reciben soporte de DB2 Spatial Extender son implantaciones de las geometrías que se muestran en la Figura 9 en la página 132. Para ver una descripción de estos tipos de datos, consulte el tema “Acerca de los tipos de datos espaciales” en la página 33.

Propiedades de geometrías y funciones asociadas

Esta sección describe las propiedades de las geometrías y las funciones espaciales asociadas a estas propiedades. La sección empieza por las propiedades principales:

- A qué clase pertenece la geometría

- Coordenadas X e Y

Esta sección también explica:

- Coordenadas Z
- Medidas
- El interior, el límite y el exterior de una geometría
- La cualidad de ser sencilla o no sencilla
- La cualidad de estar vacía o no estar vacía
- La envoltura de una geometría
- Dimensión
- El identificador del sistema de referencias espaciales asociado a una geometría

Clase

Cada geometría pertenece a una clase de la jerarquía que se muestra en la Figura 9 en la página 132. Tal como se indica en el tema "Acerca de las geometrías" en la página 131, de seis subclases de la geometría—puntos, series lineales, polígonos, varios puntos, varias series lineales y varios polígonos—se pueden crear instancias. De la superclase y de otras subclases no se pueden crear instancias.

La función ST_GeometryType toma una geometría y devuelve la subclase de la que se pueden crear instancias en el formato de una serie de caracteres. Para obtener más información, consulte el tema "ST_GeometryType" en la página 227 .

La función ST_IsValid toma una geometría asignada a un tipo de datos ST_Geometry. La función devuelve 1 (TRUE) si la geometría es válida y 0 (FALSE) si la geometría no es válida. Para obtener más información, consulte el apartado "ST_IsValid" en la página 245.

Coordenadas X e Y

Un *valor de coordenada X* indica una ubicación en relación a un punto de referencia este u oeste. Un *valor de coordenada Y* indica una ubicación en relación a un punto de referencia norte o sur. Para obtener más información, consulte los temas "La naturaleza de los datos espaciales" en la página 6 y "Acerca de los sistemas de coordenadas y de referencias espaciales" en la página 25.

Coordenadas Z

Algunas geometrías tienen una altitud o profundidad asociadas. Cada uno de los puntos que forman la geometría de una función puede incluir una coordenada Z adicional que representa una altitud o profundidad con respecto a la superficie terrestre.

La función de predicado Is3d toma una geometría y devuelve 1 (TRUE) si la función tiene coordenadas Z y 0 (FALSE) si no es así. Para obtener más información, consulte el tema “Is3d” en la página 174.

Medidas

Una medida es un valor que contiene información sobre una función geográfica y que se almacena junto con las coordenadas que definen la ubicación de la función. Por ejemplo, supongamos que está representando sistemas de transporte en su GIS. Si desea que su aplicación procese valores que indiquen distancias lineales o postes indicadores, puede almacenar estos valores junto con las coordenadas que definen las ubicaciones de los sistemas. Las medidas se almacenan como números de doble precisión.

El predicado IsMeasured toma una geometría y devuelve 1 (TRUE) si contiene medidas y 0 (FALSE) si no es así. Para obtener más información, consulte el tema “IsMeasured” en la página 175.

Interior, límite y exterior

Todas las geometrías ocupan una posición en el espacio definida por interior, límite y exterior. El exterior de una geometría es todo el espacio que no queda ocupado por la geometría. El límite de una geometría sirve como la interfaz entre su interior y su exterior. El interior es el espacio ocupado por la geometría. Las subclases heredan las propiedades de interior y exterior directamente y la propiedad de límite difiere para cada una de ellas.

La función ST_Boundary toma una geometría y devuelve una geometría que representa el límite de la geometría fuente. Para obtener más información, consulte el tema “ST_Boundary” en la página 197.

Sencilla o no sencilla

Algunas subclases de geometrías (series lineales, varios puntos y varias series lineales) pueden ser sencillas o no sencillas. Una subclase es sencilla si cumple con todas las reglas topológicas impuestas en la subclase y es no sencilla si no es así. Una serie lineal es sencilla si no forma intersección con su interior. Varios puntos es una función sencilla si ninguno de sus elementos ocupan el mismo espacio de coordenadas. Varias series lineales es una función sencilla si ninguno de los interiores de sus elementos forman intersección con su propio interior.

La función de predicado ST_IsSimple toma una geometría y devuelve 1 (TRUE) si la geometría es sencilla y 0 (FALSE) si no es así. Para obtener más información, consulte el tema “ST_IsSimple” en la página 244.

Vacía o no vacía

Una geometría está vacía si no tiene ningún punto. La envoltura, el límite, el interior y el exterior de una geometría vacía tienen el valor NULO. Una geometría vacía es siempre sencilla y puede tener coordenadas Z o medidas.

Las series lineales o varias series lineales vacías tienen una longitud 0. Los polígonos y varios polígonos vacíos tienen un área 0.

La función de predicado ST_IsEmpty toma una geometría y devuelve 1 (TRUE) si la geometría está vacía y 0 (FALSE) si no es así. Para obtener más información, consulte el tema “ST_IsEmpty” en la página 240.

Envoltura

La envoltura de una geometría es la geometría de límite formada por las coordenadas mínima y máxima (X,Y). Con las siguientes excepciones, las envolturas de la mayoría de las geometrías forman un límite en forma de rectángulo:

- La envoltura de un punto es el mismo punto, puesto que sus coordenadas mínima y máxima coinciden.
- La envoltura de una serie lineal horizontal o vertical es una serie lineal representada por el límite (los puntos finales) de la serie lineal fuente.

La función ST_Envelope toma una geometría y devuelve una geometría de límite, que representa su envoltura. Para obtener más información, consulte el tema “ST_Envelope” en la página 217.

Dimensión

Una geometría puede tener una dimensión de 0, 1 ó 2. Las dimensiones son las siguientes:

0 No tiene longitud ni área

1 Tiene longitud

2 Contiene área

Las subclases punto y varios puntos tienen una dimensión de cero. Los puntos representan funciones dimensionales que se pueden representar con una sola coordenada, mientras que las subclases de varios puntos representan datos que se deben representar con una agrupación de coordenadas desconectadas.

Las subclases serie lineal y varias series lineales tienen una dimensión de uno. Almacenan segmentos de carreteras, sistemas de ríos y afluentes y cualquier otra función cuya naturaleza sea lineal.

Las subclases polígono y varios polígonos tienen una dimensión de dos. Las funciones cuyos perímetros incluyan un área definible, como bosques, parcelas de terreno y lagos, se pueden representar mediante el tipo de datos polígono o varios polígonos.

La dimensión es importante no sólo como propiedad de la subclase, sino también para determinar la relación espacial de dos funciones. La dimensión de la función o funciones resultantes determina si la operación ha resultado o no satisfactoria. DB2 Spatial Extender examina la dimensión de las funciones para determinar el modo en que se deben comparar.

La función ST_Dimension toma una geometría y devuelve su dimensión como un entero. Para obtener más información, consulte el tema “ST_Dimension” en la página 211.

Identificador del sistema de referencias espaciales

El sistema de referencias espaciales identifica la transformación de coordenadas para cada geometría.

Se puede acceder a todos los sistemas de referencias espaciales que conoce la base de datos mediante la vista de catálogo DB2GSE.SPATIAL_REF_SYS. Para obtener información sobre esta vista, consulte el “Capítulo 11. Vistas de catálogo” en la página 117.

La función ST_SRID toma una geometría y devuelve su identificador de referencias espaciales como un entero. Para obtener más información, consulte el tema “ST_SRID” en la página 279.

La función ST_Transform asigna una geometría a un sistema de referencias espaciales que no es el sistema de referencias espaciales al que está asignada actualmente la geometría. Para obtener más información, consulte el tema “ST_Transform” en la página 284.

Geometrías con las que se pueden crear instancias y funciones asociadas

Esta sección describe las seis subclases de geometrías con las que se pueden crear instancias y las funciones que se pueden realizar sobre las mismas. Las subclases son:

- Puntos
- Series lineales
- Polígonos
- Varios puntos
- Varias series lineales
- Varios polígonos

Para ver ilustraciones de la jerarquía a la que pertenecen estas subclases y los símbolos visuales asociados a las mismas, consulte la Figura 9 en la página 132.

Puntos

Un punto es una geometría de dimensión cero que ocupa una sola ubicación en el espacio de coordenadas. Un punto incluye una coordenada X y una coordenada Y que definen esta ubicación. También puede incluir una coordenada Z y una medida.

Un punto es sencillo y tiene un límite NULO. Los puntos se suelen utilizar para definir funciones como pozos de petróleo, monumentos y elevaciones.

Funciones que se pueden aplicar únicamente a la subclase punto:

ST_Point
Toma una coordenada X, su coordenada Y asociada y el identificador del sistema de referencias espaciales al que pertenecen estas coordenadas y devuelve el punto que definen las coordenadas. Para obtener más información, consulte el tema "ST_Point" en la página 270.

ST_CoordDim
Devuelve un valor que indica qué coordenadas contiene un punto y si también contiene una medida. Este valor se denomina *dimensión de coordenadas*. Las posibles dimensiones de coordenadas son:

2 El punto consta de una coordenada X y de una coordenada Y.

3 El punto consta de una coordenada X, de una coordenada Y y de una coordenada Z.

4 El punto consta de una coordenada X, de una coordenada Y, de una coordenada Z y de una medida.

Para obtener más información, consulte el tema "ST_CoordDim" en la página 206.

ST_PointFromText
Toma una representación de texto en binario conocida (WKT) de OGC de un punto y devuelve el punto. Para obtener más información, consulte el tema "ST_PointFromText" en la página 267.

ST_X Devuelve un valor de coordenada X del tipo de datos ST_Point como un número de doble precisión. Para obtener más información, consulte el tema "ST_X" en la página 290.

ST_Y Devuelve un valor de coordenada Y del tipo de datos ST_Point como un número de doble precisión. Para obtener más información, consulte el tema "ST_Y" en la página 291.

Z Devuelve un valor de coordenada Z del tipo de datos ST_Point como un número de doble precisión. Para obtener más información, consulte el tema "Z" en la página 292.

M Devuelve una medida del tipo de datos ST_Point como un número de doble precisión. Para obtener más información, consulte el tema “M” en la página 182.

Series lineales

Una serie lineal es un objeto de una dimensión que se almacena como una secuencia de puntos que definen una ruta lineal interpolada. La serie lineal es sencilla si no forma intersección con su interior. Los puntos finales (el límite) de una serie lineal cerrada ocupan el mismo punto en el espacio. Una serie lineal es un anillo si es cerrada y si su interior no forma intersección consigo mismo. Además de las otras propiedades heredades de la geometría de superclase, las series lineales tienen longitud. Las series lineales se suelen utilizar para definir funciones lineales como carreteras, ríos y cables de alta tensión.

Una serie lineal sencilla cuyos punto inicial y punto final coinciden se denomina un *anillo*.

Los puntos finales suelen formar el límite de una serie lineal a no ser que esta esté cerrada, en cuyo caso el límite es NULO. El interior de una serie lineal es una ruta conectada que queda dentro de los puntos finales a no ser que esté cerrada, en cuyo caso el interior es continuo.

Funciones que se aplican a series lineales:

ST_StartPoint
Toma una serie lineal y devuelve su primer punto. Para obtener más información, consulte el tema “ST_StartPoint” en la página 280.

ST_EndPoint
Toma una serie lineal y devuelve su último punto. Para obtener más información, consulte el tema “ST_Endpoint” en la página 216.

ST_PointN
Toma una serie lineal y un índice al punto *n* y devuelve dicho punto. Para obtener más información, consulte el tema “ST_PointN” en la página 271.

ST_Length
Toma una serie lineal y devuelve su longitud como un número de doble precisión. Para obtener más información, consulte el tema “ST_Length” en la página 247.

ST_NumPoints
Toma una serie lineal y devuelve el número de puntos de su secuencia como un entero. Para obtener más información, consulte el tema “ST_NumPoints” en la página 262.

ST_IsRing

Toma una serie lineal y devuelve 1 (TRUE) si la serie lineal es un anillo y 0 (FALSE) si no lo es. Para obtener más información, consulte el tema “ST_IsRing” en la página 242.

ST_IsClosed

Toma una serie lineal y devuelve 1 (TRUE) si la serie lineal está cerrada y 0 (FALSE) si no es así. Para obtener más información, consulte el tema “ST_IsClosed” en la página 238.

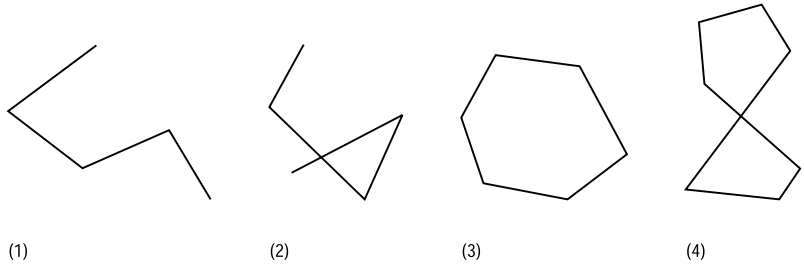


Figura 10. Objetos de serie lineal.

1. Serie lineal sencilla no cerrada.
2. Serie lineal no sencilla no cerrada.
3. Serie lineal sencilla cerrada, por lo tanto anillo.
4. Serie lineal no sencilla cerrada. No es un anillo.

Polígonos

Un polígono es una superficie de dos dimensiones que se almacena como una secuencia de puntos que definen su anillo límite exterior y 0 o más anillos interiores. Los anillos de un polígono no se pueden solapar. Por lo tanto, por definición, los polígonos son siempre sencillo. Suelen definir parcelas de terreno, lagos y otras funciones que tienen una extensión espacial.

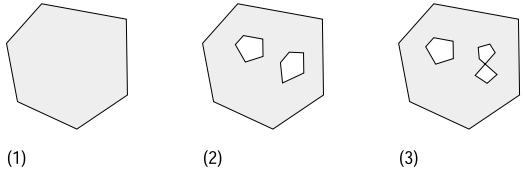


Figura 11. Polígonos.

1. Un polígono cuyo límite está definido por un anillo exterior.
2. Un polígono cuyo límite está definido por un anillo exterior y dos anillos interiores. El área que queda dentro de los anillos interiores forma parte del exterior del polígono.
3. Un polígono válido porque los anillos forman intersección en un solo punto tangente.

El exterior y los anillos interiores definen el límite de un polígono y el espacio comprendido entre los anillos define el interior del polígono. Los anillos de un polígono pueden formar intersección en un punto tangente, pero no se pueden cruzar. Además de las otras propiedades heredadas de la geometría de superclase, los polígonos tienen área.

Funciones que se aplican a polígonos:

ST_Area
Toma un polígono y devuelve su área como un número de doble precisión. Para obtener más información, consulte el tema "ST_Area" en la página 193.

ST_ExteriorRing
Toma un polígono y devuelve su anillo exterior como una serie lineal. Para obtener más , consulte el tema "ST_ExteriorRing" en la página 220.

ST_NumInteriorRing
Toma un polígono de devuelve el número de anillos interiores que contiene. Para obtener más información, consulte el tema "ST_NumInteriorRing" en la página 261.

ST_InteriorRingN
Toma un polígono y un índice y devuelve el anillo interior número *n* como una serie lineal. Para obtener más información, consulte el tema "ST_InteriorRingN" en la página 229.

ST_Centroid
Toma un polígono y devuelve un punto que es el centro de la extensión del polígono. Para obtener más información, consulte el tema "ST_Centroid" en la página 201.

ST_PointOnSurface
Toma un polígono y devuelve un punto que seguro que está en la superficie del polígono. Para obtener más información, consulte el tema "ST_PointOnSurface" en la página 272.

ST_Perimeter
Toma un polígono y devuelve el perímetro de su superficie. Para obtener más información, consulte el tema "ST_Perimeter" en la página 266.

Varios puntos

La geometría varios puntos es un grupo de puntos y, al igual que sus elementos, tiene una dimensión 0. La geometría varios puntos es sencilla si ninguno de sus elementos ocupa el mismo espacio de coordenadas. El límite de una geometría varios puntos es NULO. Se pueden utilizar varios puntos para definir fenómenos como patrones de emisiones aéreas e incidentes de un brote epidémico.

Funciones que se aplican a las geometrías varios puntos:

ST_NumGeometries
Toma un grupo homogéneo y devuelve el número de elementos de la geometría base que contiene. Para obtener más información, consulte el tema “ST_NumGeometries” en la página 260.

ST_GeometryN
Toma un grupo homogéneo y un índice y devuelve la geometría base número n. Para obtener más información, consulte el tema “ST_GeometryN” en la página 226.

Varias series lineales

Una geometría varias series lineales es un grupo de series lineales. Las geometrías varias series lineales son sencillas si forman intersección únicamente en los puntos finales de los elementos de series lineales. Las geometrías varias series lineales son no sencillas si los interiores de los elementos series lineales forman intersección.

El límite de una geometría varias series lineales son los puntos finales que no forman intersección de los elementos series lineales. La geometría varias series lineales está cerrada si todos sus elementos series lineales están cerrados. El límite de una geometría varias series lineales es NULO si todos los puntos finales de todos sus elementos forman intersección. Además de las otras propiedades heredadas de la geometría de superclase, las geometrías varias series lineales tienen longitud. Las geometrías de varias series lineales se utilizan para definir corrientes o redes de carreteras.

Funciones que se aplican a varias series lineales:

ST_Length
Toma una geometría varias series lineales y devuelve la longitud acumulada de todos sus elementos series lineales como un número de doble precisión. Para obtener más información, consulte el tema “ST_Length” en la página 247.

ST_IsClosed
Toma una geometría varias series lineales y devuelve 1 (TRUE) si la geometría está cerrada y 0 (FALSE) si no es así. Para obtener más información, consulte el tema “ST_IsClosed” en la página 238.

ST_NumGeometries
Toma un grupo homogéneo y devuelve el número de elementos de la geometría base que contiene. Para obtener más información, consulte el tema “ST_NumGeometries” en la página 260.

ST_GeometryN

Toma un grupo homogéneo y un índice y devuelve la geometría base número n. Para obtener más información, consulte el tema “ST_GeometryN” en la página 226.

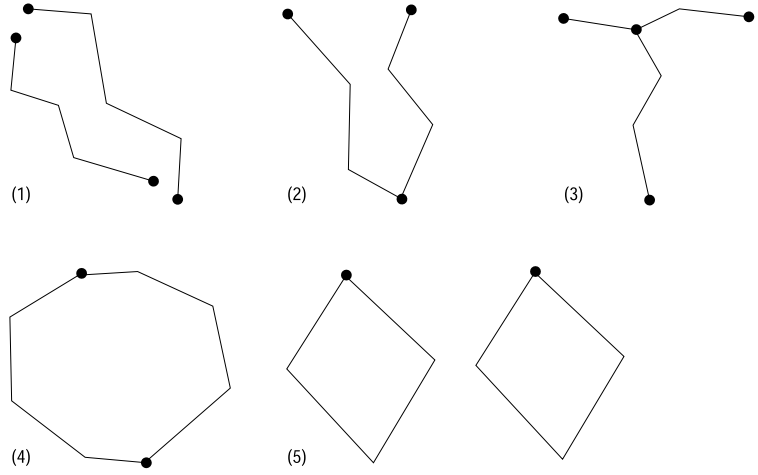


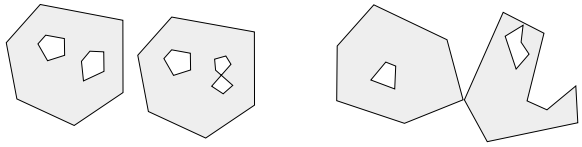
Figura 12. Varias series lineales.

1. Una geometría varias series lineales sencilla cuyo límite está definido por los cuatro puntos finales de sus dos elementos series lineales.
2. Una geometría varias series lineales sencilla porque sólo los puntos finales de los elementos series lineales forman intersección. El límite queda definido por los dos puntos finales que no forman intersección.
3. Una geometría varias series lineales no sencilla porque el interior de uno de sus elementos series lineales forma intersección. El límite de esta geometría varias series lineales queda definido por los cuatro puntos finales, incluido el punto de intersección.
4. Una geometría varias series lineales sencilla no cerrada. No está cerrada porque sus elementos series lineales no están cerrados. Es sencilla porque ninguno de los interiores de ninguno de sus elementos series lineales forman intersección.
5. Una geometría varias series lineales sencilla cerrada. Está cerrada porque todos sus elementos están cerrados. Es sencilla porque ninguno de sus elementos forma intersección en sus interiores.

Varios polígonos

El límite de una geometría varios polígonos es la longitud acumulada de los anillos exteriores e interiores de sus elementos. El interior de una geometría varios polígonos se define como los interiores acumulados de sus elementos polígonos. El límite de los elementos de una geometría varios polígonos puede formar intersección únicamente en un punto tangente. Además de las otras propiedades heredadas de la geometría de superclase, las geometrías varios polígonos tienen área. Las geometrías varios polígonos definen

funciones como grupos de bosques o una parcela de terreno no continua, como una cadena de islas.



(1) (2)

Figura 13. Varios polígonos.

1. Una geometría varios polígonos con dos elementos polígonos. El límite queda definido por los dos anillos exteriores y los tres anillos interiores.
2. Una geometría varios polígonos con dos elementos polígonos. El límite queda definido por los dos anillos exteriores y los dos anillos interiores. Los dos elementos polígonos forman intersección en un punto tangente.

Funciones que se aplican a varias las geometrías varios polígonos:

ST_Area
Toma una geometría varios polígonos y devuelve el área acumulada de sus elementos polígonos como un número de doble precisión. Para obtener más información, consulte el tema “ST_Area” en la página 193.

ST_Centroid
Toma una geometría varios polígonos y devuelve un punto que es su centro geométrico ponderado. Para obtener más información, consulte el tema “ST_Centroid” en la página 201.

ST_NumGeometries
Toma un grupo homogéneo y devuelve el número de elementos de la geometría base que contiene. Para obtener más información, consulte el tema “ST_NumGeometries” en la página 260.

ST_GeometryN
Toma un grupo homogéneo y un índice y devuelve la geometría base número n. Para obtener más información, consulte el tema “ST_GeometryN” en la página 226.

Funciones que muestran relaciones y comparaciones, generan geometrías y convierten formatos de valores

Las secciones anteriores contenían una introducción a las tres categorías de funciones espaciales:

- Funciones asociadas a propiedades de las geometrías
- Funciones asociadas a determinadas geometrías

Esta sección contiene una introducción a otras tres categorías:

- Funciones que determinan formas en que las funciones geográficas se pueden relacionar o comparar
- Funciones que generan geometrías nuevas
- Funciones que convierten los valores de una geometría en un formato que se puede importar o exportar

Funciones que muestran relaciones o comparaciones entre funciones geográficas

Varias funciones espaciales devuelven información sobre formas en que las funciones geográficas se pueden relacionar o comparar entre sí. La mayoría de estas funciones, denominadas *predicados*, son funciones booleanas. Esta sección describe los predicados en general y luego cada uno de ellos por separado.

Funciones de predicado

Las funciones de predicado devuelven 1 (TRUE) si una comparación cumple con los criterios de la función o 0 (FALSE) si la comparación falla. Los predicados que comprueban una relación espacial comparan pares de geometrías que pueden ser de distinto tipo o dimensión.

Los predicados comparan las coordenadas X e Y de las geometrías sometidas. Las coordenadas Z y la medida (si las hay) se pasan por alto. Esto permite comparar geometrías que tienen coordenadas Z o medida con otras que no las tienen.

El *Modelo de intersección 9 ampliado dimensionalmente (DE-9IM*) [1]es un enfoque matemático que define la relación espacial entre pares de geometrías de distintos tipos y dimensiones. Este modelo expresa las relaciones espaciales entre todos los tipos de geometrías como intersecciones entre pares de su interior, límite y exterior, teniendo en cuenta la dimensión de las intersecciones resultantes.

Supongamos que tenemos las geometrías *a* y *b*: I(*a*), B(*a*) y E(*a*) representan el interior, límite y exterior de *a*. E I(b), B(b) y E(b) representan el interior, límite y exterior de *b*. La intersección de I(a), B(a) y E(a) con I(b), B(b) y E(b) genera una matriz 3 por 3. Cada intersección puede dar lugar a geometrías de

1. El DE-91M fue desarrollado por Clementini y Felice, quienes ampliaron dimensionalmente el Modelo de intersección 9 de Egenhofer y Herring. DE-91M es fruto de la colaboración entre cuatro autores, Clementini, Eliseo, Di Felice y van Osstrom. Publicaron el modelo en ″A Small Set of Formal Topological Relationships Suitable for End-User Interaction,″ D. Abel y B.C. Ooi (Ed.), *Advances in Spatial Database—Third International Symposium. SSD '93*. LNCS 692. Págs. 277-295. El Modelo de intersección 9 de Springer-Verlag Singapore (1993) Egenhofer M.J. y Herring, J., se publicó en ″Categorizing binary topological relationships between regions, lines, and points in geographic databases,″ *Tech. Report, Department of Surveying Engineering*, Universidad de Maine, Orono, ME 1991.

diferentes dimensiones. Por ejemplo, la intersección de los límites de dos polígonos consta de un punto y una serie lineal, en cuyo caso la función dim devolvería la dimensión máxima de 1.

La función dim devuelve un valor de 1, 0, 1 ó 2. El 1 corresponde a la serie nula de dim(null), que se devuelve cuando no se encuentra ninguna intersección.

	Interior	**Límite**	**Exterior**
Interior	dim(I(a) ∩ I(b))	dim(I(a) ∩ B(b))	dim(I(a) ∩ E(b))
Límite	dim(B(a) ∩ I(b))	dim(B(a) ∩ B(b))	dim(B(a) ∩ E(b))
Exterior	dim(E(a) ∩ I(b))	dim(E(a) ∩ B(b))	dim(E(a) ∩ E(b))

Los resultados de los predicados de relación espacial se pueden entender o comprobar comparando los resultados del predicado con una matriz patrón que representa los valores aceptables para el DE-9IM.

La matriz patrón contiene los valores aceptables para cada una de las celtas de la matriz de intersecciones. Los posibles valores patrón son:

T Debe existir una intersección, dim = 0, 1 ó 2.

F No debe existir intersección, dim = -1.

***** No importa si existe o no intersección, dim = -1, 0, 1 ó 2.

0 Debe existir una intersección y su dimensión máxima debe ser 0, dim = 0.

1 Debe existir una intersección y su dimensión máxima debe ser 1, dim = 1.

2 Debe existir una intersección y su dimensión máxima debe ser 2, dim = 2.

Por ejemplo, la siguiente matriz patrón correspondiente al predicado ST_Within incluye los valores T, F y *.

Tabla 40. Matriz correspondiente a ST_Within. La matriz patrón del predicado ST_Within correspondiente a combinaciones de geometrías.

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	T	*	F
	Límite	*	*	F
	Exterior	*	*	*

El predicado ST_Within devuelve TRUE si los interiores de ambas geometrías forman intersección y si el interior y el límite de *a* no forma intersección con el exterior de *b*. Las demás condiciones no importan.

Cada predicado tiene al menos una matriz patrón, pero algunos necesitan más de una para describir las relaciones de combinaciones de varios tipos de geometrías.

ST_Equals

ST_Equals devuelve 1 (TRUE) si las dos geometrías del mismo tipo tienen valores de coordenadas X,Y idénticos.

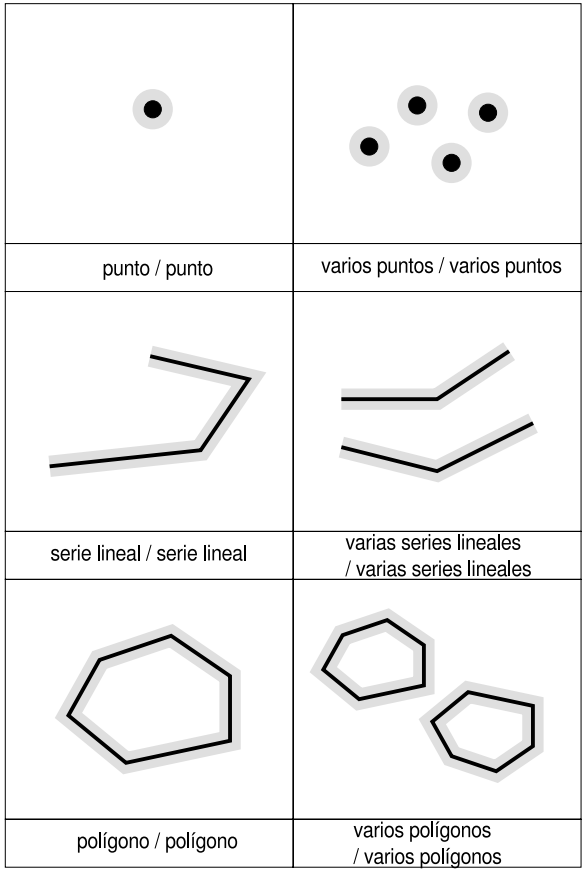


Figura 14. ST_Equals. Las geometrías son iguales si tienen coordenadas X,Y coincidentes.

Tabla 41. Matriz correspondiente a igualdad. La matriz de patrón DE-9IM correspondiente a igualdad asegura que los interiores forman intersección y que ninguna parte del interior o del límite de ninguna geometría forma intersección con el exterior de la otra.

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	T	*	F
	Límite	*	*	F
	Exterior	F	F	*

Para obtener más información, consulte el tema “ST_Equals” en la página 219.

ST_OrderingEquals

ST_OrderingEquals compara dos geometrías y devuelve 1 (TRUE) si las geometrías son iguales y las coordenadas están en el mismo orden; si no es así, devuelve 0 (FALSE). Para obtener más información, consulte el tema “ST_OrderingEquals” en la página 263.

ST_Disjoint

ST_Disjoint devuelve 1 (TRUE) si la intersección de las dos geometrías es un conjunto vacío.

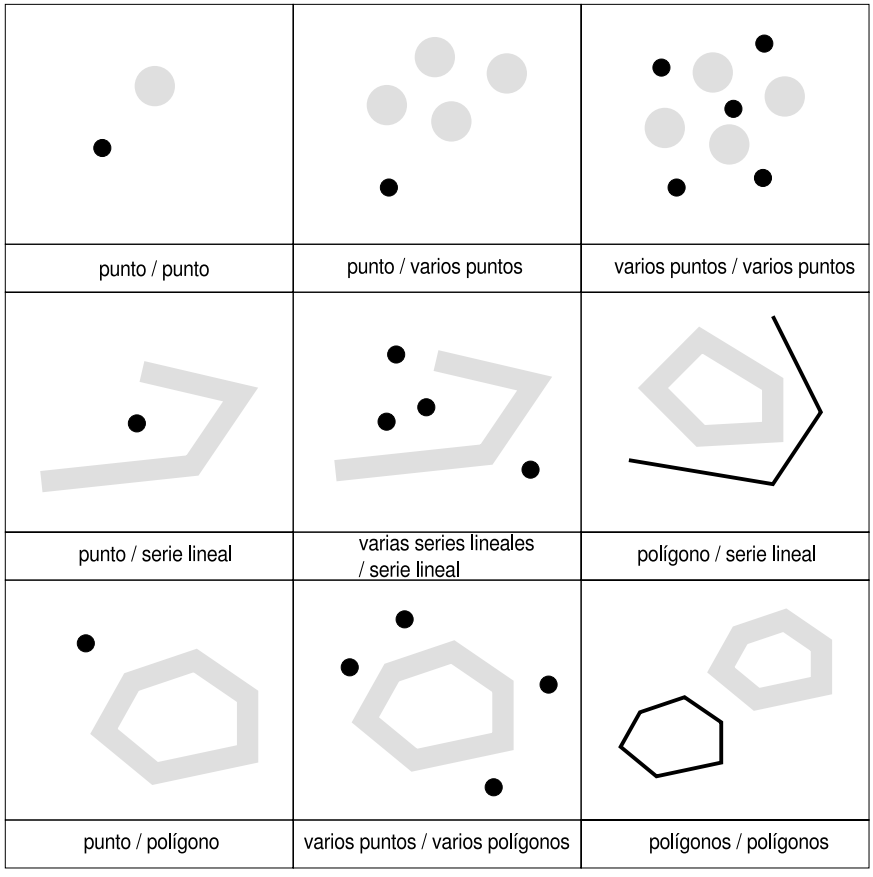


Figura 15. ST_Disjoint. Las geometrías están desunidas si no forman intersección entre sí en ninguna forma.

Tabla 42. Matriz correspondiente a ST_Disjoint. La matriz patrón del predicado ST_Disjoint indica que ni los interiores ni los límites de ninguna geometría forman intersección.

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	F	F	*
	Límite	F	F	*
	Exterior	*	*	*

Para obtener más información, consulte el tema “ST_Disjoint” en la página 213.

ST_Intersects

ST_Intersects devuelve 1 (TRUE) si la intersección no da como resultado un conjunto vacío. La intersección devuelve exactamente el resultado opuesto a ST_Disjoint.

El predicado ST_Intersects devuelve TRUE si las condiciones de algunas de las siguientes matrices patrón devuelven TRUE.

Tabla 43. Matriz correspondiente a ST_Intersects (1). El predicado ST_Intersects devuelve TRUE si los interiores de ambas geometrías forman intersección.

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	T	*	*
	Límite	*	*	*
	Exterior	*	*	*

Tabla 44. Matriz correspondiente a ST_Intersects (2). El predicado ST_Intersects devuelve TRUE si el límite de la primera geometría forma intersección con el límite de la segunda.

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	*	T	*
	Límite	*	*	*
	Exterior	*	*	*

Tabla 45. Matriz correspondiente a ST_Intersects (3). El predicado ST_Intersects devuelve TRUE si el límite de la primera geometría forma intersección con el interior de la segunda.

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	*	*	*
	Límite	T	*	*
	Exterior	*	*	*

Tabla 46. Matriz correspondiente a ST_Intersects (4). El predicado ST_Intersects devuelve TRUE si los límites de alguna de las geometrías forman intersección.

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	*	*	*
	Límite	*	T	*
	Exterior	*	*	*

Para obtener más información, consulte el tema “ST_Intersects” en la página 237.

EnvelopesIntersect

Esta función devuelve 1 (TRUE) su las envolturas de dos geometrías forman intersección. Es una función muy útil que implanta de forma eficiente ST_Intersects (`ST_Envelope(g1),ST_Envelope(g2)`). Para obtener más información, consulte el tema “EnvelopesIntersect” en la página 172.

ST_Touches

ST_Touches devuelve 1 (TRUE) si ninguno de los puntos comunes a ambas geometrías forman intersección con los interiores de ambas geometrías. Al menos una geometría debe ser serie lineal, polígono, varias series lineales o varios polígonos.

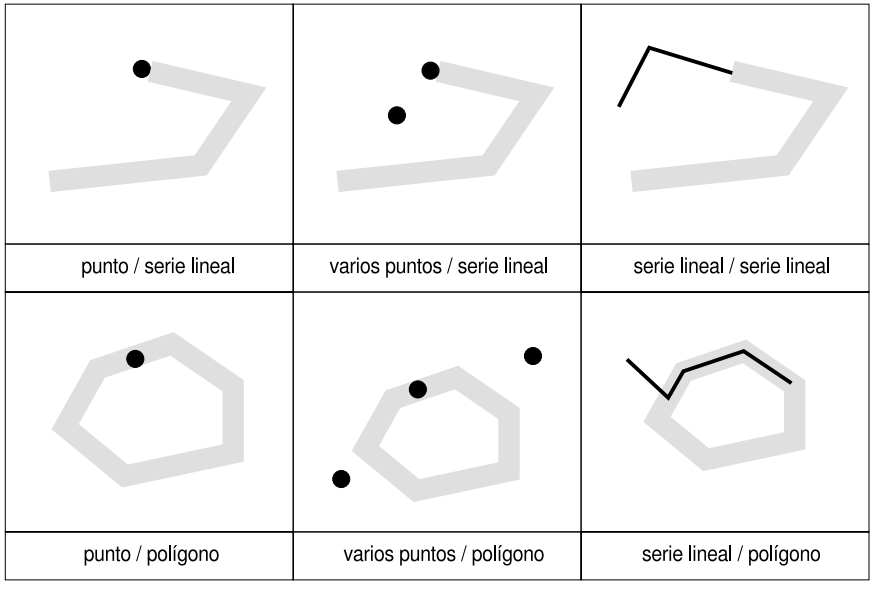


Figura 16. ST_Touches

Las matrices patrón muestran que el predicado ST_Touches devuelve TRUE cuando los interiores de las geometrías no forman intersección y el límite de alguna de las geometrías forma intersección con el interior o el límite de la otra.

Tabla 47. Matriz correspondiente a ST_Touches (1)

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	F	T	*
	Límite	*	*	*
	Exterior	*	*	*

Tabla 48. Matriz correspondiente a ST_Touches (2)

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	F	*	*
	Límite	T	*	*
	Exterior	*	*	*

Tabla 49. Matriz correspondiente a ST_Touches (3)

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	F	*	*
	Límite	*	T	*
	Exterior	*	*	*

Para obtener más información, consulte el tema “ST_Touches” en la página 283.

ST_Overlaps

ST_Overlaps compara dos geometrías de la misma dimensión. Devuelve 1 (TRUE) si su conjunto de intersecciones da lugar a una geometría distinta de ambas, pero que tenga la misma dimensión.

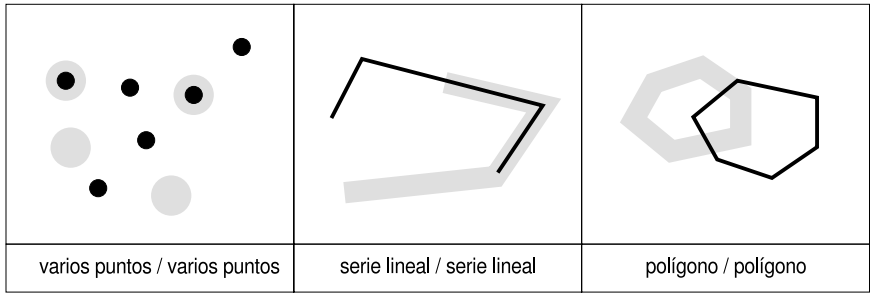


Figura 17. ST_Overlaps

La matriz patrón de la Tabla 50 se aplica a recubrimientos polígono / polígono, varios puntos / varios puntos y varios polígonos / varios polígonos. Para estas combinaciones, el predicado overlay devuelve TRUE si el interior de cada geometría forma intersección con el interior y el exterior del otro.

Tabla 50. Matriz correspondiente a ST_Overlaps (1)

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	T	*	T
	Límite	*	*	*
	Exterior	T	*	*

La matriz patrón de la Tabla 51 se aplica a recubrimientos serie lineal / serie lineal y varias series lineales / varias series lineales. En este caso, la intersección de las geometrías debe dar lugar a una geometría con una dimensión de 1 (otra serie lineal). Si la dimensión de la intersección de los interiores es 1, el predicado ST_Overlaps devolvería FALSE, pero el predicado ST_Crosses devolvería TRUE.

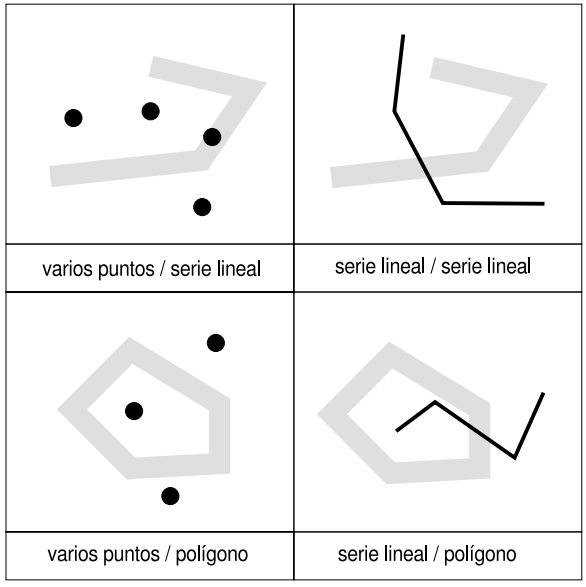
Tabla 51. Matriz correspondiente a ST_Overlaps (2)

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	1	*	T
	Límite	*	*	*
	Exterior	T	*	*

Para obtener más información, consulte el tema “ST_Overlaps” en la página 264.

ST_Crosses

ST_Crosses devuelve 1 (TRUE) si la intersección da lugar a una geometría cuya dimensión es un grado menor que la dimensión máxima de las dos geometrías fuente y el conjunto de intersecciones es interior a ambas geometrías fuente. ST_Crosses devuelve 1 (TRUE) sólo para comparaciones varios puntos / polígono, varios puntos / serie lineal, serie lineal / serie lineal, serie lineal / polígono y serie lineal / varios polígonos.



La matriz patrón de la Tabla 52 se aplica a varios puntos / serie lineal, varios puntos / varias series lineales, varios puntos / polígono, varios puntos / varios polígonos, serie lineal / polígono, serie lineal / varios polígonos. La matriz indica que los interiores deben formar intersección y que el interior de la geometría principal (geometría *a*) debe formar intersección con el exterior de la secundaria (geometría *b*).

Tabla 52. Matriz correspondiente a ST_Crosses (1)

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	T	*	T
	Límite	*	*	*
	Exterior	*	*	*

La matriz patrón de la Tabla 53 en la página 155 se aplica a serie lineal / serie lineal, serie lineal / varias series lineales y varias series lineales / varias series lineales. La matriz indica que la dimensión de la intersección de los interiores debe ser 0 (intersección en un punto). Si la dimensión de esta intersección es 1 (intersección en una serie lineal), el predicado ST_Crosses devuelve FALSE; sin embargo, el predicado ST_Overlaps devuelve TRUE.

Tabla 53. Matriz correspondiente a ST_Crosses (2)

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	0	*	*
	Límite	*	*	*
	Exterior	*	*	*

Para obtener más información, consulte el tema “ST_Crosses” en la página 208.

ST_Within

ST_Within devuelve 1 (TRUE) si la primera geometría queda completamente dentro de la segunda geometría. ST_Within devuelve exactamente el resultado opuesto a ST_Contains.

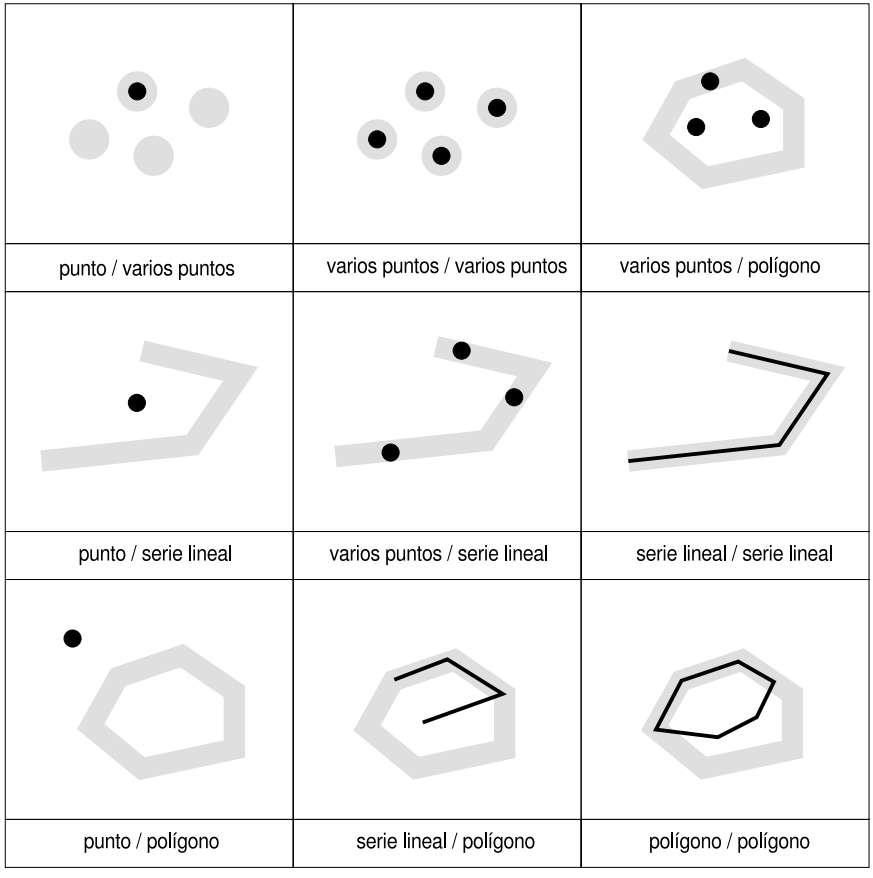


Figura 18. Within

La matriz patrón del predicado ST_Within indica que los interiores de ambas geometrías deben formar intersección, y que el interior y el límite de la geometría principal (geometría *a*) no deben formar intersección con el exterior de la secundaria (geometría *b*).

Tabla 54. Matriz correspondiente a ST_Within

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	T	*	F
	Límite	*	*	F
	Exterior	*	*	*

Para obtener más información, consulte el tema “ST_Within” en la página 286.

ST_Contains

ST_Contains devuelve 1 (TRUE) si la segunda geometría queda completamente contenida en la primera geometría. El predicado ST_Contains devuelve exactamente el resultado opuesto al predicado ST_Within.

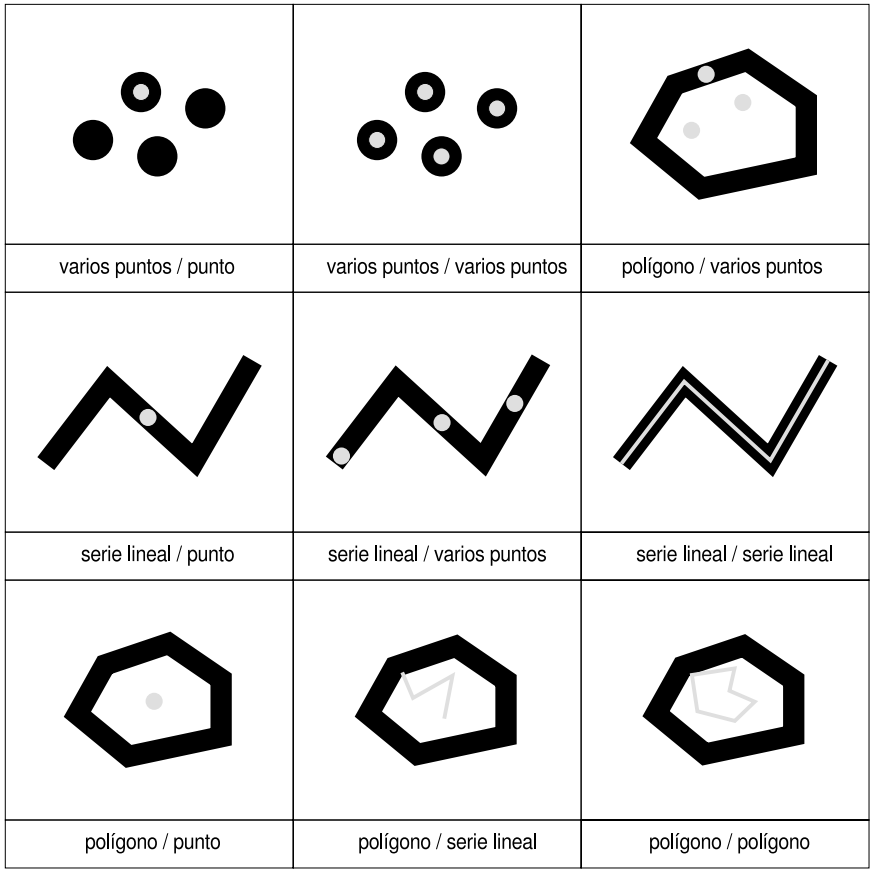


Figura 19. ST_Contains

La matriz patrón del predicado ST_Contains indica que los interiores de ambas geometrías deben forman intersección y que el interior y el límite de la geometría secundaria (geometría *b*) no debe formar intersección con el exterior de la principal (geometría *a*).

Tabla 55. Matriz correspondiente a ST_Contains

		b		
		Interior	**Límite**	**Exterior**
a	**Interior**	T	*	*
	Límite	*	*	*
	Exterior	F	F	*

Para obtener más información, consulte el tema “ST_Contains” en la página 202.

ST_Relate

La función ST_Relate compara dos geometrías y devuelve 1 (TRUE) si las geometrías cumplen con las condiciones especificadas por la serie de la matriz patrón DE-91M; si no es así, la función devuelve 0 (FALSE). Para obtener más información, consulte el tema “ST_Relate” en la página 277.

ST_Distance

La función ST_Distance indica la distancia mínima que separa dos funciones desunidas. Si las funciones no están desunidas, la función indicará una distancia mínima de 0.

Por ejemplo, ST_Distance podría indicar la distancia más corta que debe recorrer un avión entre dos ubicaciones. La Figura 20 ilustra esta información.

Figura 20. Distancia mínima entre dos ciudades. ST_Distance puede tomar las coordenadas correspondientes a las ciudades de Los Ángeles y Chicago como entrada y devolver un valor que indique la distancia mínima entre estas ubicaciones.

Para obtener más información, consulte el tema “ST_Distance” en la página 215.

Funciones que generan geometrías nuevas a partir de geometrías existentes

DB2 Spatial Extender ofrece predicados y funciones de transformación que generan geometrías nuevas a partir de geometrías existentes.

ST_Intersection

La función ST_Intersection devuelve el conjunto de intersecciones de dos geometrías. El conjunto de intersecciones siempre se devuelve como un grupo que es la dimensión mínima de las geometrías fuente. Por ejemplo, para una serie lineal que forma intersección con un polígono, la función de intersección devuelve una geometría varias series lineales compuesta de la parte de la serie lineal común al interior y al límite del polígono. La geometría varias series lineales contiene más de una serie lineal si la serie lineal fuente forma

intersección con el polígono en dos o más segmentos discontinuos. Si las geometrías no forman intersección o si la intersección da como resultado una dimensión inferior a las de las geometrías fuente, se devuelve una geometría vacía.

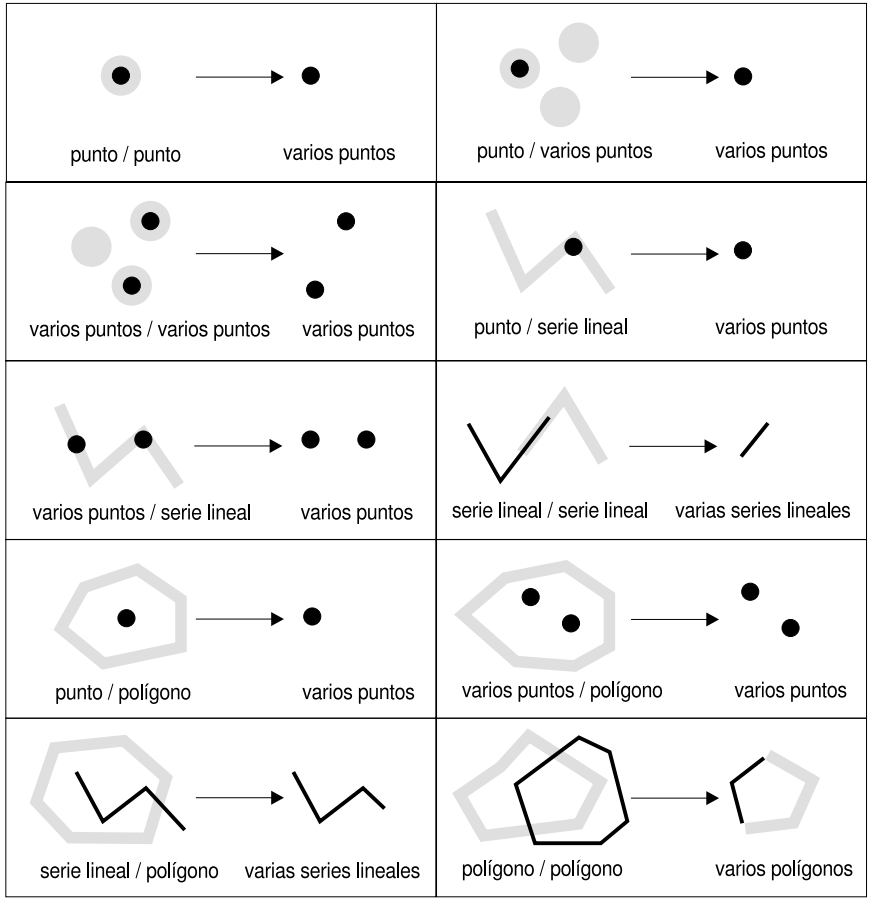


Figura 21. ST_Intersection. Ejemplos de la función ST_Intersection.

Para obtener más información, consulte el tema “ST_Intersection” en la página 235.

ST_Difference

La función ST_Difference devuelve la parte de la geometría principal que no forma intersección con la geometría secundaria. Se trata del AND NOT lógico de espacio. La función ST_Difference sólo funciona sobre geometrías de la misma dimensión y devuelve un grupo que tiene la misma dimensión que las

geometrías fuente. En el caso de que las geometrías fuente fueran iguales, se devolvería una geometría vacía.

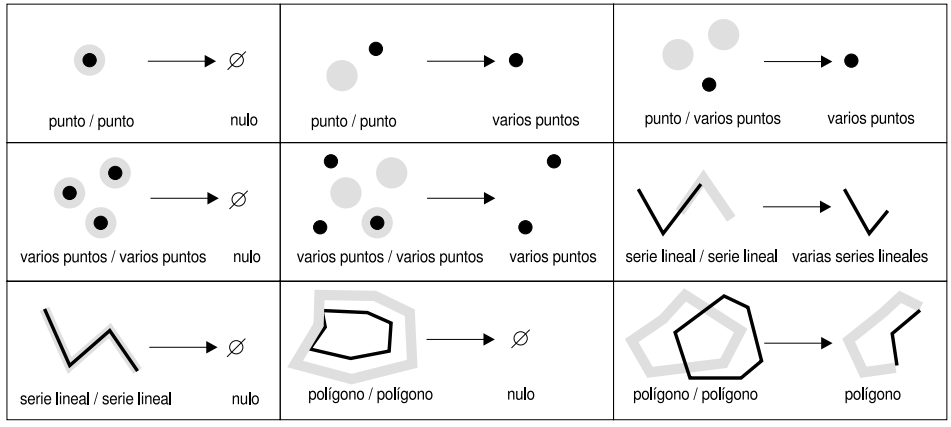


Figura 22. ST_Difference

Para obtener más información, consulte el tema “ST_Difference” en la página 210.

ST_Union

La función ST_Union devuelve el conjunto de unión de dos geometrías. Es el OR lógico de espacio. Las geometrías fuente deben tener la misma dimensión. ST_Union siempre devuelve el resultado como un grupo.

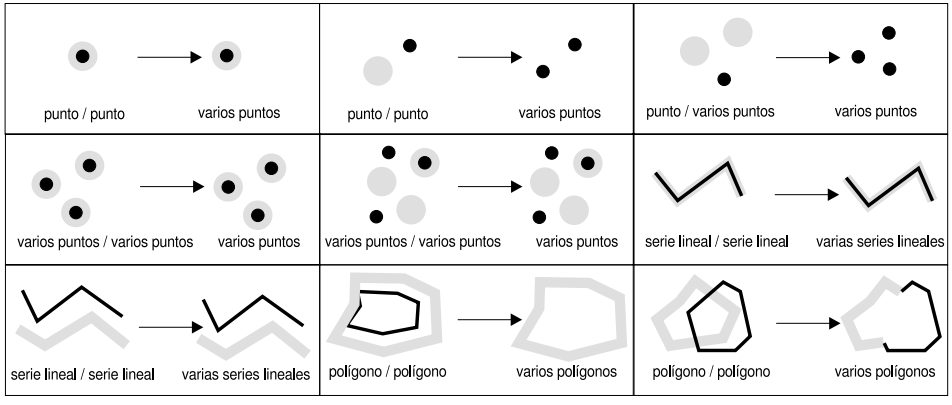


Figura 23. ST_Union

Para obtener más información, consulte el tema “ST_Union” en la página 285.

ST_Buffer

La función ST_Buffer genera una geometría rodeando una geometría a una distancia especificada. Da como resultado un polígono cuando se rodea una geometría principal o cuando los elementos de un grupo están lo suficientemente juntos para que todos los polígonos rodeados se solapan. Sin embargo, si hay suficiente separación entre los elementos de un grupo rodeado, se obtendrá como resultado polígonos individuales rodeados; en este caso la función ST_Buffer devuelve una geometría varios polígonos.

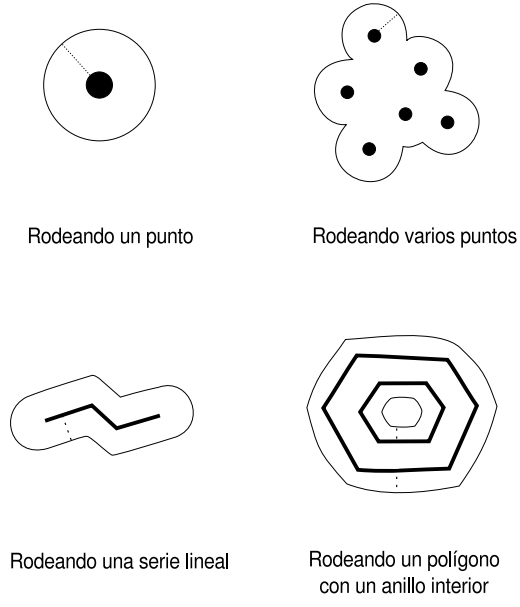


Figura 24. ST_Buffer

La función ST_Buffer acepta distancias positivas y negativas; sin embargo, sólo las geometrías con una dimensión de dos (polígonos y varios polígonos) aplican un rodeo negativo. El valor absoluto de la distancia rodeada se utiliza cuando la dimensión de la geometría fuente es menor que 2 (todas las geometrías que no son polígono o varios polígonos).

En general, para anillos exteriores, las distancias de rodeo positivas generan anillos de polígono que están lejos del centro de la geometría fuente; las distancias de rodeo negativas generan anillos de polígono o de varios polígonos hacia el centro. Para anillos interiores de un polígono o de varios polígonos, una distancia de rodeo positiva genera un anillo de rodeo hacia el centro y una distancia de rodeo negativa genera un anillo de rodeo alejado del centro.

El proceso de rodeo fusiona polígonos que se solapan. Las distancias negativas superiores a la mitad de la anchura interior máxima de un polígono dan como resultado una geometría vacía.

Para obtener más información, consulte el tema “ST_Buffer” en la página 199.

LocateAlong

Para geometrías que tienen medidas, la ubicación de una determinada medida se puede encontrar con la función LocateAlong. LocateAlong devuelve la ubicación como una geometría varios puntos. Si la dimensión de la geometría fuente es 0 (por ejemplo, un punto y una geometría varios puntos), se necesita una coincidencia exacta y los puntos que tienen un valor de medida coincidente se devuelven como una geometría varios puntos. Sin embargo, para geometrías fuente cuya dimensión es mayor que 0, la ubicación se interpola. Por ejemplo, si el valor de medida entrado es 5,5 y las medidas de los vértices de una serie lineal son 3, 4, 5, 6 y 7 respectivamente, se devuelve el punto interpolado que queda exactamente a mitad de camino entre los vértices con los valores de medida 5 y 6.

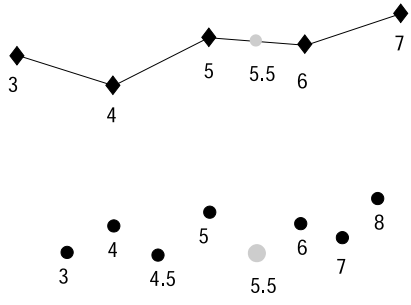


Figura 25. LocateAlong

Para obtener más información, consulte el tema “LocateAlong” en la página 178.

LocateBetween

La función LocateBetween devuelve una serie de rutas o ubicaciones que quedan entre dos valores de medida de una geometría fuente que tiene medidas. Si la dimensión de la geometría fuente es 0, LocateBetween devuelve una geometría varios puntos que contiene todos los puntos cuyas medidas quedan entre las dos medidas fuente. Para geometrías fuente cuya dimensión es mayor que 0, LocateBetween devuelve una geometría varias series lineales si se puede interpolar una ruta; si no es así, LocateBetween devuelve una geometría varios puntos que contiene las ubicaciones de los puntos. Se devuelve un punto vacío si LocateBetween no puede interpolar una ruta ni encontrar ninguna ubicación entre las medidas. LocateBetween realiza una búsqueda tipo inclusive de las geometrías; por lo tanto, las medidas de las

geometrías deben ser superiores o iguales que la medida *origen* e inferiores o iguales que la medida *destino*.

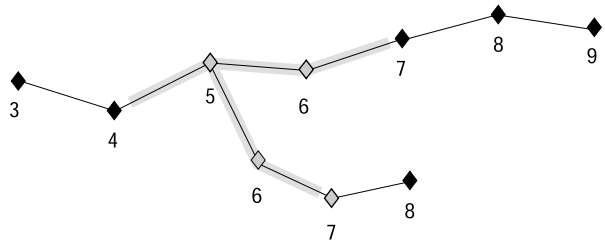


Figura 26. LocateBetween

Para obtener más información, consulte el tema “LocateBetween” en la página 180.

ST_ConvexHull

La función ST_ConvexHull devuelve el polígono de casco convexo de cualquier geometría que tenga al menos tres vértices que formen una forma convexa. Si los vértices de la geometría no forman una forma convexa, ST_ConvexHull devuelve un nulo. ST_ConvexHull suele ser el primer paso del proceso utilizado para crear una red TIN a partir de una serie de puntos.

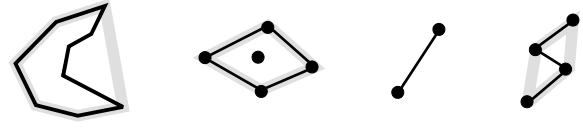


Figura 27. ST_ConvexHull

Para obtener más información, consulte el tema “ST_ConvexHull” en la página 204.

ST_Polygon

Genera un polígono a partir de una serie lineal. Para obtener más información, consulte el tema “ST_Polygon” en la página 276.

Funciones que convierten el formato de los valores de una geometría

DB2 Spatial Extender da soporte a tres formatos de intercambio de datos GIS:

- Representación de texto conocido
- Representación de binario conocido
- Representación de forma binaria ESRI

Representación de texto conocido

DB2 Spatial Extender tiene varias funciones que generan geometrías a partir de descripciones de texto.

ST_WKTToSQL
Crea una geometría a partir de una representación de texto de cualquier tipo de geometría. No se debe especificar ningún identificador de sistema de referencias espaciales. Para obtener más información, consulte el tema "ST_WKTToSQL" en la página 289.

ST_GeomFromText
Crea una geometría a partir de una representación de texto de cualquier tipo de geometría. Se debe especificar un identificador de sistema de referencias espaciales. Para obtener más información, consulte el tema "ST_GeometryFromText" en la página 222.

ST_PointFromText
Crea un punto a partir de una representación de texto de punto. Para obtener más información, consulte el tema "ST_PointFromText" en la página 267.

ST_LineFromText
Crea una serie lineal a partir de una representación de texto de serie lineal. Para obtener más información, consulte el tema "ST_LineFromText" en la página 249.

ST_PolyFromText
Crea un polígono a partir de una representación de texto de polígono. Para obtener más información, consulte el tema "ST_PolyFromText" en la página 273.

ST_MPointFromText
Crea una geometría varios puntos a partir de una representación de varios puntos. Para obtener más información, consulte el tema "ST_MPointFromText" en la página 255.

ST_MLineFromText
Crea una geometría varias series lineales a partir de una representación de varias series lineales. Para obtener más información, consulte el tema "ST_MLineFromText" en la página 252.

ST_MPolyFromText
Crea una geometría varios polígonos a partir de una representación de varios polígonos. Para obtener más información, consulte el tema "ST_MPolyFromText" en la página 258.

La representación de texto es una serie ASCII. Permite intercambiar una geometría en formato de texto ASCII. Estas funciones no necesitan la

definición de ninguna estructura de programa especial para correlacionar una representación binaria. Por lo tanto, se pueden utilizar en un programa 3GL o 4GL.

La función ST_AsText convierte un valor de geometría existente en una representación de texto. Para obtener más información, consulte el tema "ST_AsText" en la página 196.

Para ver una descripción detallada de representaciones de texto conocido, consulte el tema "Las representaciones de texto conocido OGC" en la página 303.

Representación de binario conocido

DB2 Spatial Extender tiene varias funciones que generan geometrías a partir de representaciones de binario conocido (WKB).

ST_WKBToSQL
Crea una geometría a partir de una representación de binario conocido de cualquier tipo de geometría. No se debe especificar ningún identificador de sistema de referencias espaciales. Para obtener más información, consulte el tema "ST_WKBToSQL" en la página 287.

ST_GeomFromWKB
Crea una geometría a partir de una representación de binario conocido de cualquier tipo de geometría. Se debe especificar un identificador de sistema de referencias espaciales. Para obtener más información, consulte el tema "ST_GeomFromWKB" en la página 224.

ST_PointFromWKB
Crea un punto a partir de una representación de binario conocido de un punto. Para obtener más información, consulte el tema "ST_PointFromWKB" en la página 268.

ST_LineFromWKB
Crea una serie lineal a partir de una representación de binario conocido de una serie lineal. Para obtener más información, consulte el tema "ST_LineFromWKB" en la página 250.

ST_PolyFromWKB
Crea un polígono a partir de una representación de binario conocido de un polígono. Para obtener más información, consulte el tema "ST_PolyFromWKB" en la página 274.

ST_MPointFromWKB
Crea una geometría varios puntos a partir de una representación de binario conocido de una geometría varios puntos. Para obtener más información, consulte el tema "ST_MPointFromWKB" en la página 256.

ST_MLineFromWKB
Crea una geometría varias series lineales a partir de una representación de binario conocido de una geometría varias series lineales. Para obtener más información, consulte el tema “ST_MLineFromWKB” en la página 253.

ST_MPolyFromWKB
Crea una geometría varios polígonos a partir de una representación de binario conocido de una geometría varios polígonos. Para obtener más información, consulte el tema “ST_MPolyFromWKB” en la página 259.

La representación de binario conocido es una corriente continua de bytes. Permite intercambiar una geometría entre un cliente ODBC y una base de datos SQL en formato binario. Estas funciones geométricas necesitan la definición de estructuras C para correlacionar la representación binaria. Por lo tanto, están destinadas a ser utilizadas dentro de un programa 3GL y no se ajustan a un entorno 4GL.

La función ST_AsBinary convierte un valor de geometría existente en una representación de binario conocido. Para obtener más información, consulte el tema “ST_AsBinary” en la página 195.

Para ver una descripción detallada de las representaciones de binario conocido, consulte el tema “Las representaciones de binario conocido (WKB) de OGC” en la página 308.

Representación de forma ESRI
DB2 Spatial Extender tiene varias funciones que generan geometrías a partir de una representación de forma ESRI. La representación de forma ESRI da soporte a coordenadas Z y a medidas, además de las representaciones de dos dimensiones soportadas por las representaciones de binario conocido y de texto.

ShapeToSQL
Crea una geometría a partir de una forma de cualquier tipo de geometría. No se debe especificar ningún identificador de sistema de referencias espaciales. Para obtener más información, consulte el tema “ShapeToSQL” en la página 191.

GeometryFromShape
Crea una geometría a partir de una forma de cualquier tipo de geometría. Se debe especificar un identificador de sistema de referencias espaciales. Para obtener más información, consulte el tema “GeometryFromShape” en la página 171.

PointFromShape
Crea un punto a partir de una forma de punto. Para obtener más información, consulte el tema “PointFromShape” en la página 187.

LineFromShape
Crea una serie lineal a partir de una forma de línea poligonal. Para obtener más información, consulte el tema “LineFromShape” en la página 176.

PolyFromShape
Crea un polígono a partir de una forma de línea poligonal. Para obtener más información, consulte el tema “PolyFromShape” en la página 189.

MPointFromShape
Crea una geometría varios puntos a partir de una forma de varios puntos. Para obtener más información, consulte el tema “MPointFromShape” en la página 185.

MLineFromShape
Crea una geometría varias series lineales a partir de una forma de línea poligonal de varias partes. Para obtener más información, consulte el tema “MLine FromShape” en la página 183.

MPolyFromShape
Crea una geometría varios polígonos a partir de una forma de polígono de varias partes. Para obtener más información, consulte el tema “MPolyFromShape” en la página 186.

La sintaxis general de estas funciones es la misma. El primer argumento es la representación de forma que se entra como un tipo de datos BLOB. El segundo argumento es el identificador de referencias espaciales que se asignará a la geometría. Por ejemplo, la función GeometryFromShape tiene la siguiente sintaxis:

```
GeometryFromShape(shapegeometry, SRID)
```

Para correlacionar la representación binaria, estas funciones de forma necesitan la definición de estructuras C. Por lo tanto, están destinadas a ser utilizadas dentro de un programa 3GL y no se ajustan a un entorno 4GL.

La función AsBinaryShape convierte un valor de geometría en una representación de forma ESRI. Para obtener más información, consulte el tema “AsBinaryShape” en la página 170.

Para ver una descripción detallada de las representaciones de forma, consulte el tema “Las representaciones de forma ESRI” en la página 312.

Capítulo 14. Funciones espaciales para consultas SQL

Este capítulo contiene las funciones disponibles que sirven para consultar datos espaciales. Cada función se describe en una sección que le muestra la sintaxis, el tipo de retorno y ejemplos de código. Algunos de los ejemplos de este capítulo incluyen una sentencia CREATE TABLE en la que hay una o más columnas definidas como columnas espaciales.

Las siguientes consideraciones se aplican a las funciones espaciales:

- Los ejemplos de este capítulo están calificados con el nombre de biblioteca db2gse. En lugar de calificar de forma explícita cada función y tipo espacial con db2gse, puede definir la vía de acceso a la función para que incluya db2gse.
- Antes de insertar datos en una columna espacial:
  - Es posible que tenga que aumentar el parámetro udf_mem_sz. El valor recomendado es 2048. Si 2048 no resulta adecuado, aumente el parámetro udf_mem_sz en incrementos de 256.
  - Debe registrarlo como una capa. Para obtener más información sobre cómo registrar una columna espacial como una capa, consulte “Capítulo 4. Cómo definir columnas espaciales, registrarlas como capas y habilitar un geocodificador para que las mantenga” en la página 33.

AsBinaryShape

AsBinaryShape toma un objeto de geometría y devuelve un BLOB.

Sintaxis

db2gse.AsBinaryShape(g db2gse.ST_Geometry)

Tipo devuelto

BLOB(1m)

Ejemplos

El siguiente fragmento de código ilustra el modo en que la función AsBinaryShape convierte los polígonos de zona de la tabla SENSITIVE_AREAS en polígonos de forma. Estos polígonos de forma se pasan a la función draw_polygon de la aplicación para que los muestre.

```
/* Crear la expresión SQL. */
strcpy(sqlstmt, "select db2gse.AsBinaryShape (zone) from SENSITIVE_AREAS
where db2gse.EnvelopesIntersect(zone, db2gse.PolyFromShape(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Preparar la sentencia SQL. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* Definir la longitud pcbvalue1 de la forma. */
pcbvalue1 = blob_len;

/* Vincular el parámetro de forma. */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, SQL_BLOB, blob_len,
0, shape, blob_len, &pcbvalue1);

/* Ejecutar la consulta. */
rc = SQLExecute (hstmt);

/* Asignar los resultados de la consulta, (los polígonos de zona) a la
   variable fetched_binary. */
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);

/* Buscar y cargar cada polígono dentro de la ventana de visualización y mostrarlo. */

while(SQL_SUCCESS == (rc = SQLFetch(hstmt)))
  draw_polygon(fetched_binary);
```

GeometryFromShape

GeometryFromShape toma una forma y una identidad del sistema de referencias espaciales para devolver un objeto de geometría.

Sintaxis

db2gse.GeometryFromShape(ShapeGeometry Blob(1M), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

El siguiente fragmento de código C contiene funciones ODBC intercaladas con funciones SQL de DB2 Spatial Extender que insertan datos en la tabla LOTS.

La tabla LOTS se creó con dos columnas: la columna LOT_ID, que identifica de forma exclusiva cada parcela, y la columna de polígonos LOT, que contiene la geometría de cada parcela.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot     db2gse.ST_MultiPolygon);
```

La función GeometryFromShape convierte formas en geometría de DB2 Spatial Extender. La sentencia INSERT completa se copia en shp_sql. La sentencia INSERT contiene marcadores de parámetro para aceptar los datos LOT_ID y los datos LOT de forma dinámica.

```
/* Crear la sentencia SQL insert para llenar el id de parcela y
  los varios polígonos de parcela. Los interrogantes son marcadores de parámetro
  que indican los valores de lot_id y lot que se recuperarán en el momento
  de la ejecución. */
strcpy (shp_sql,"insert into LOTS (lot_id, lot) values (?,
db2gse.GeometryFromShape (cast(? as blob(1m)), db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Vincular el valor de clave de entero con el primer parámetro. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Vincular la forma con el segundo parámetro. */
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

EnvelopesIntersect

EnvelopesIntersect devuelve 1 (TRUE) si las envolturas de las dos geometrías forman intersección y 0 (FALSE) si no es así.

Sintaxis

db2gse.EnvelopesIntersect(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

La función get_window recupera las coordenadas de la ventana de visualización a partir de la aplicación. El parámetro de ventana es realmente una estructura de forma de polígono que contiene una serie de coordenadas que representan el polígono de visualización. La función PolyFromShape convierte la forma de ventana de visualización en un polígono de DB2 Spatial Extender que la función EnvelopesIntersect utiliza como su envoltura de intersección. Se devuelven todos los polígonos de la zona SENSITIVE_AREAS que forman intersección con el interior o el límite de la ventana de visualización. Cada polígono se busca y carga del conjunto de resultados y se pasa a la función draw_polygon.

```
/* Obtener las coordenadas de la ventana de visualización
como una forma de polígono.
get_window(&window)

/* Crear la expresión SQL. Se utilizará la función db2gse.EnvelopesIntersect
  para limitar el conjunto de resultados a los polígonos de zona que formen
  intersección con la envoltura de la ventana de visualización. */
strcpy(sqlstmt, "select db2gse.AsBinaryShape(zone) from SENSITIVE_AREAS where
db2gse.EnvelopesIntersect (zone, db2gse.PolyFromShape(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Definir blob_len en la longitud de bytes de un polígono de forma de
5 puntos. */
blob_len = 128;

/* Preparar la sentencia SQL. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* Definir pcbvalue1 en la forma de la ventana */
pcbvalue1 = blob_len;

/* Vincular el parámetro de forma. */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, SQL_BLOB,
blob_len,0, window, blob_len, &pcbvalue1);

/* Ejecutar la consulta. */
rc = SQLExecute (hstmt);

/* Asignar los resultados de la consulta, (los polígonos de zona) a la
   variable fetched_binary. */
```

```
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);

/* Buscar y cargar cada polígono dentro de la ventana de visualización
y mostrarlo. */
while(SQL_SUCCESS == (rc = SQLFetch(hstmt))
  draw_polygon(fetched_binary);
```

Is3d

Is3d toma un objeto de geometría y devuelve 1 (TRUE) si el objeto tiene coordenadas 3D y 0 (FALSE) si no es así.

Sintaxis

db2gse.Is3d(g db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla THREED_TEST, que tiene dos columnas, la columna GID de tipo entero y la columna de geometría G1.

```
CREATE TABLE THREED_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Las sentencias INSERT insertan dos puntos en la tabla THREED_TEST. El primer punto no contiene coordenadas Z y el segundo sí.

```
INSERT INTO THREED_TEST
VALUES(1, db2gse.ST_PointFromText('point (10 10)', db2gse.coordref()..srid(0)))

INSERT INTO THREED_TEST
VALUES (2, db2gse.ST_PointFromText('point z (10.92 10.12 5)',
db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT lista el contenido de la columna GID con los resultados de la función Is3d. La función devuelve un 0 para la primera fila, que no contiene coordenadas Z y un 1 para la segunda fila, que sí contiene coordenadas Z.

```
SELECT gid, db2gse.Is3d (g1) "Is it 3d?" FROM THREED_TEST
```

Se devuelve el siguiente conjunto de resultados.

```
gid      Is it 3d?
------   ----------
     1            0
     2            1
```

IsMeasured

IsMeasured toma un objeto de geometría y devuelve 1 (TRUE) si el objeto tiene medidas y 0 (FALSE) si no es así.

Sintaxis

db2gse.IsMeasured(g db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla MEASURE_TEST, que tiene dos columnas. La columna GID identifica de forma exclusiva las filas y la columna G1 almacena las geometrías de punto.

```
CREATE TABLE MEASURE_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Las siguientes sentencias INSERT insertan dos registros en la tabla MEASURE_TEST. El primer registro almacena un punto que no tiene medida. El punto del segundo registro sí tiene una medida.

```
INSERT INTO MEASURE_TEST
VALUES(1, db2gse.ST_PointFromText('point (10 10)', db2gse.coordref()..srid(0)))

INSERT INTO MEASURE_TEST
VALUES (2, db2gse.ST_PointFromText('point m (10.92 10.12 5)',
    db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente muestran la columna GID junto con los resultados de la función IsMeasured. La función IsMeasured devuelve un 0 para la primera fila porque el punto no tiene medida. Devuelve un 1 para la segunda fila porque el punto tiene medidas.

```
SELECT gid, db2gse.IsMeasured (g1) "Has measures?" FROM MEASURE_TEST
gid      Has measures
------   ----------
     1            0
     2            1
```

LineFromShape

LineFromShape toma una forma de tipo punto y una identidad del sistema de referencias espaciales y devuelve una serie lineal.

Sintaxis

db2gse.Line FromShape(ShapeLineString Blob(1M), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_LineString

Ejemplos

El siguiente fragmento de código llena la tabla SEWERLINES con el id exclusivo, clase de tamaño y geometría de cada línea de alcantarillado.

La sentencia CREATE TABLE crea la tabla SEWERLINES, que tiene tres columnas. La primera columna, SEWER_ID, identifica de forma exclusiva cada línea de alcantarillado. La segunda columna, CLASS, de tipo entero, identifica el tipo de línea de alcantarillado, que suele estar asociado a la capacidad de la línea. La tercera columna, SEWER, de tipo serie lineal, almacena la geometría de la línea de alcantarillado.

```
CREATE TABLE SEWERLINES (sewer_id   integer, class  integer,
                         sewer db2gse.ST_LineString);

/* Crear la sentencia SQL insert para llenar el id de alcantarillado,
   la clase de tamaño y la serie lineal del alcantarillado.
   Los interrogantes son marcadores de parámetro que indican los valores
   de id de alcantarillado, clase y geometría de alcantarillado que se
   recuperarán en el momento de la ejecución. */
strcpy (shp_sql,"insert into sewerlines (sewer_id,class,sewer)
values (?,?, db2gse.Line FromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Vincular el valor de clave de entero con el primer parámetro. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_id, 0, &pcbvalue1);

/* Vincular el valor entero class con el segundo parámetro. */
pcbvalue2 = 0;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_class, 0, &pcbvalue2);;

/* Vincular la forma con el tercer parámetro. */
pcbvalue3 = blob_len;
```

```
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, sewer_shape, blob_len, &pcbvalue3);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

LocateAlong

LocateAlong toma un objeto de geometría y una medida para devolver como geometría varios puntos el grupo de puntos encontrados en la medida.

Sintaxis

db2gse.LocateAlong(g db2gse.ST_Geometry, adistance Double)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla LOCATEALONG_TEST. LOCATEALONG_TEST tiene dos columnas: la columna GID, que identifica de forma exclusiva cada fila, y la columna de geometría G1, que almacena la geometría de ejemplo.

```
CREATE TABLE LOCATEALONG_TEST (gid integer, g1 db2gse.ST_Geometry)
```

Las siguientes sentencias INSERT insertan dos filas. La primera es una geometría varias series lineales; la segunda es una geometría varios puntos.

```
INSERT INTO db2gse.LOCATEALONG_TEST VALUES(
1, db2gse.ST_MLineFromText('multilinestring m ((10.29 19.23 5,23.82 20.29 6,
                                        30.19 18.47 7,
45.98 20.74 8), (23.82 20.29 6,30.98 23.98 7,42.92 25.98 8))',
                                  db2gse.coordref()..srid(0)))

INSERT INTO db2gse.LocateAlong_TEST VALUES(
2, db2gse.ST_MPointFromText('multipoint m (10.29 19.23 5,23.82 20.29 6,
 30.19 18.47 7,45.98 20.74 8,23.82 20.29 6,30.98 23.98 7,42.92 25.98)',
                 db2gse.coordref()..srid(0)))
```

En la siguiente sentencia SELECT y el correspondiente conjunto de resultados, se indica a la función LocateAlong que busque puntos cuya medida es 6,5. La primera fila devuelve una geometría varios puntos que contiene dos puntos. Sin embargo, la segunda fila devuelve un punto vacío. Para funciones lineales (geometría con una dimensión mayor que 0), LocateAlong puede interpolar el punto; sin embargo, para varios puntos, la medida de destino debe coincidir exactamente.

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.LocateAlong (g1,6.5)) AS
varchar(96))
"Geometry"
FROM LOCATEALONG_TEST

GID         Geometry
----------- ----------------------------------------------------------------

          1 MULTIPOINT M ( 27.01000000 19.38000000 6.50000000, 27.40000000
```

```
22.14000000 6.50000000)
          2 POINT EMPTY

  2 record(s) selected.
```

En la siguiente sentencia SELECT y el correspondiente conjunto de resultados, la función LocateAlong devuelve una geometría varios puntos para ambas filas. La medida de destino de 7 coincide con las medidas de los datos fuente de geometría varias series lineales y de geometría varios puntos.

```
SELECT gid,CAST(db2gse.ST_AsText(db2gse.LocateAlong (g1,7)) AS varchar(96))
   "Geometry"
FROM LOCATEALONG_TEST

GID         Geometry

----------- ----------------------------------------------------------------
          1 MULTIPOINT M ( 30.19000000 18.47000000 7.00000000, 30.98000000
 23.98000000 7.00000000)
          2 MULTIPOINT M ( 30.19000000 18.47000000 7.00000000, 30.98000000
 23.98000000 7.00000000)

  2 record(s) selected.
```

LocateBetween

LocateBetween toma un objeto de geometría y dos ubicaciones de medidas y devuelve una geometría que representa el grupo de rutas desconectadas entre las dos ubicaciones de medidas.

Sintaxis

db2gse.LocateBetween(g db2gse.ST_Geometry, adistance Double, anotherdistance Double)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla LOCATEBETWEEN_TEST. LOCATEBETWEEN_TEST tiene dos columnas: la columna GID, que identifica de forma exclusiva cada fila, y la columna de varias series lineales G1, que almacena la geometría de ejemplo.

```
CREATE TABLE LOCATEBETWEEN_TEST (gid integer, g1 db2gse.ST_Geometry)
```

Las siguientes sentencias INSERT insertan dos filas en la tabla LOCATEBETWEEN_TEST. La primera fila es una geometría varias series lineales y la segunda es una geometría varios puntos.

```
INSERT INTO db2gse.LOCATEBETWEEN_TEST
VALUES(1,db2gse.ST_MLineFromText('multilinestring m ((10.29 19.23 5,
                          23.82 20.29 6, 30.19 18.47 7,45.98 20.74 8),
                                        (23.82 20.29 6,30.98 23.98 7,
                                         42.92 25.98 8))',
                      db2gse.coordref()..srid(0)))

INSERT INTO db2gse.LOCATEBETWEEN_TEST
VALUES(2, db2gse.ST_MPointFromText('multipoint m (10.29 19.23 5,23.82 20.29 6,
30.19 18.47 7,45.98 20.74 8,23.82 20.29 6,
30.98 23.98 7,42.92 25.98 8)',
                        db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente muestran cómo la función LocateBetween localiza las medidas que quedan entre las medidas 6,5 y 7,5, ambas inclusive. La primera fila devuelve una geometría varias series lineales que contiene varias series lineales. La segunda fila devuelve varios puntos porque los datos fuente eran varios puntos. Cuando los datos fuente tienen una dimensión de 0 (punto o varios puntos), se necesita una coincidencia exacta.

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.LocateBetween (g1,6.5,7.5))
       AS varchar(96)) "Geometry"
FROM LOCATEBETWEEN_TEST

GID         Geometry
----------- ----------------------------------------------------------------
          1 MULTILINESTRING M ( 27.01000000 19.38000000 6.50000000, 31.19000000
18.47000000 7.00000000,38.09000000 19.61000000 7.50000000),(27.40000000 22.1400
0000 6.50000000, 30.98000000 23.98000000 7.00000000,36.95000000 24.98000000 7.5
0000000)
          2 MULTIPOINT M ( 30.19000000 18.47000000 7.00000000, 30.98000000 23.9
8000000 7.00000000)

  2 record(s) selected.
```

M

M toma un punto y devuelve su medida.

Sintaxis

db2gse.M(p db2gse.ST_Point)

Tipo devuelto

Double

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla M_TEST. M_TEST tiene dos columnas: la columna de entero GID, que identifica de forma exclusiva la fila, y la columna de punto PT1, que almacena la geometría de ejemplo.

```
CREATE TABLE M_TEST (gid integer, pt1 db2gse.ST_Point)
```

Las siguientes sentencias INSERT insertan una fila que contiene un punto con medidas y una fila que contiene un punto sin medidas.

```
INSERT INTO db2gse.M_TEST
VALUES(1, db2gse.ST_PointFromText('point (10.02 20.01)',
db2gse.coordref()..srid(0)))

INSERT INTO db2gse.M_TEST
VALUES(2, db2gse.ST_PointFromText('point  zm(10.02 20.01 5.0 7.0)',
db2gse.coordref()..srid(0)))
```

En la siguiente sentencia SELECT y el conjunto de resultados correspondiente, la función M lista los valores de medida de los puntos. Puesto que el primer punto no tiene medidas, la función M devuelve un NULO.

```
SELECT gid, db2gse.M (pt1) "The measure" FROM M_TEST

GID         The measure
----------- ------------------------
          1                        -
          2   +7.00000000000000E+000

  2 record(s) selected.
```

MLine FromShape

MLineFromShape toma una forma de tipo varias series lineales y una identidad del sistema de referencias espaciales y devuelve una geometría varias series lineales.

Sintaxis

db2gse.MLineFromShape(ShapeMultiLineString Blob(1M), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_MultiLineString

Ejemplos

El siguiente fragmento de código llena la tabla WATERWAYS con un id exclusivo, un nombre y una geometría varias series lineales denominada water.

La tabla WATERWAYS se crea con las columnas ID y NAME que identifican cada sistema de corrientes y de ríos almacenado en la tabla. La columna WATER es una geometría varias series lineales porque los sistemas de ríos y de corrientes suelen ser un agregado de varias series lineales.

```
CREATE TABLE WATERWAYS (id     integer,
                        name       varchar(128),
                        water      db2gse.ST_MultiLineString);

/* Crear la sentencia SQL insert para llenar el id, nombre y
   varias series lineales. Los interrogantes son marcadores de parámetro que
   indican los valores de id, name y water que se recuperarán en el
   momento de la ejecución. */
strcpy (shp_sql,"insert into WATERWAYS (id,name,water)
values (?,?, db2gse.MLineFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Vincular el valor entero id con el primer parámetro. */

pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);
/* Vincular el valor varchar name con el segundo parámetro. */

pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, name_len, 0, &name, name_len, &pcbvalue2);

/* Vincular la forma con el tercer parámetro. */
```

```
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, water_shape, blob_len, &pcbvalue3);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

MPointFromShape

MPointFromShape toma una forma de tipo varios puntos y una identidad del sistema de referencias espaciales para devolver una geometría varios puntos.

Sintaxis

db2gse.MPointFromShape(ShapeMultiPoint (1M), srs db2gse.coordref)

Tipo devuelto

db2gse.ST_MultiPoint

Ejemplos

Este fragmento de código llena una tabla SPECIES_SITINGS de biología.

La tabla SPECIES_SITINGS se crea con tres columnas. Las columnas species y genus identifican de forma exclusiva cada fila mientras que la geometría varios puntos sitings almacena las ubicaciones de las localizaciones de especies.

```
CREATE TABLE SPECIES_SITINGS (species  varchar(32),
                              genus varchar(32),
                              sitings  db2gse.ST_MultiPoint);

/* Crear la sentencia SQL insert para llenar las columnas species, genus y
   sitings. Los interrogantes son marcadores de parámetro que indican los
   valores de name y water que se recuperarán durante la ejecución.*/
strcpy (shp_sql,"insert into SPECIES_SITINGS (species,genus,sitings)
values (?,?, db2gse.MPointFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Vincular el valor varchar species con el primer parámetro. */
pcbvalue1 = species_len;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, species_len, 0, species, species_len, &pcbvalue1);

/* Vincular el valor varchar genus con el segundo parámetro. */
pcbvalue2 = genus_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, genus_len, 0, name, genus_len, &pcbvalue2);

/* Vincular la forma con el tercer parámetro. */
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, sitings_len, 0, sitings_shape, sitings_len, &pcbvalue3);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

MPolyFromShape

MPolyFromShape toma una forma de tipo varios polígonos y una identidad del sistema de referencias espaciales para devolver una geometría varios polígonos.

Sintaxis

db2gse.MPolyFromShape(ShapeMultiPolygon Blob(1m), srs db2gse.coordref)

Tipo devuelto

db2gse.ST_MultiPolygon

Ejemplos

Este fragmento de código llena la tabla LOTS.

La tabla LOTS almacena los id de parcela que identifican de forma exclusiva cada parcela y la geometría varios polígonos de parcela que contiene la geometría de líneas de la parcela.

```
CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );

/* Crear la sentencia SQL insert para llenar las columnas lot_id y lot. Los
   interrogantes son marcadores de parámetro que indican los valores lot_id y lot
   que se recuperarán en el momento de la ejecución.*/
strcpy (shp_sql,"insert into LOTS (lot_id,lot)
values (?, db2gse.MPolyFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Vincular el valor entero lot_id con el primer parámetro. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Vincular la forma de parcela con el segundo parámetro. */
pcbvalue2 = lot_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, lot_len, 0, lot_shape, lot_len, &pcbvalue2);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

PointFromShape

PointFromShape toma una forma de tipo punto y una identidad del sistema de referencias espaciales para devolver un punto.

Sintaxis

db2gse.PointFromShape(db2gse.ShapePoint blob(1M), srs db2gse.coordref)

Tipo devuelto

db2gse.ST_Point

Ejemplos

El fragmento del programa llena la tabla HAZARDOUS_SITES.

Los sitios peligrosos se almacenan en la tabla HAZARDOUS_SITES, creada con la siguiente sentencia CREATE TABLE. La columna location, definida como un punto, almacena una ubicación que representa el centro geográfico de cada sitio peligroso.

```
CREATE TABLE HAZARDOUS_SITES ( site_id   integer,
                               name      varchar(128),
                               location  db2gse.ST_Point);

/* Crear la sentencia SQL insert para llenar las columnas site_id, name y
   location. Los interrogantes son marcadores de parámetro que indican los
   valores de site_id, name y location que se recuperarán en el momento de
   la ejecución. */
strcpy (shp_sql,"insert into HAZARDOUS_SITES (site_id, name, location)
values (?,?, db2gse.PointFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Vincular el valor entero site_id con el primer parámetro. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);

/* Vincular el valor varchar name con el segundo parámetro. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Vincular la forma location con el tercer parámetro. */
pcbvalue3 = location_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
        SQL_BLOB, location_len, 0, location_shape, location_len, &pcbvalue3);

    /* Ejecutar la sentencia insert. */
    rc = SQLExecute (hstmt);
```

PolyFromShape

PolyFromShape toma una forma de tipo polígono y una identidad del sistema de referencias espaciales para devolver un polígono.

Sintaxis

db2gse.PolyFromShape (ShapePolygon Blob(1M), srs db2gse.coordref)

Tipo devuelto

db2gse.ST_Polygon

Ejemplos

El fragmento del programa llena la tabla SENSITIVE_AREAS. Los interrogantes representan marcadores de parámetro para los valores id, name, size, type y zone que se recuperarán en el momento de la ejecución.

La tabla SENSITIVE_AREAS contiene varias columnas que describen las instituciones amenazadas además de la columna zone, que almacena la geometría polígono de la institución.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                               size     float,
                               type     varchar(10),
                               zone     db2gse.ST_Polygon);

/* Crear la sentencia SQL insert para llenar las columnas
   id, name, size, type y zone. Los interrogantes son marcadores
   de parámetro que indican los valores id, name, size, type y zone
   que se recuperarán en el momento de la ejecución. */
strcpy (shp_sql,"insert into SENSITIVE_AREAS (id, name, size, type, zone)
values (?,?,?,?, db2gse.PolyFromShape (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Vincular el valor entero id con el primer parámetro. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);
/* Vincular el valor varchar name con el segundo parámetro. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Vincular el valor float size con el tercer parámetro. */
pcbvalue3 = 0;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_FLOAT,
```

```
     SQL_REAL, 0, 0, &size, 0, &pcbvalue3);

/* Vincular el valor varchar type con el cuarto parámetro. */
pcbvalue4 = type_len;
rc = SQLBindParameter (hstmt, 4, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_VARCHAR, type_len, 0, type, type_len, &pcbvalue4);

/* Vincular el polígono zone con el quinto parámetro. */
pcbvalue5 = zone_len;
rc = SQLBindParameter (hstmt, 5, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, zone_len, 0, zone_shp, zone_len, &pcbvalue5);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

ShapeToSQL

ShapeToSQL construye un valor db2gse.ST_Geometry a partir de su representación de binario conocido. El valor SRID de 0 se utiliza automáticamente.

Sintaxis

db2gse.ST_ShapeToSQL(ShapeGeometry blob(1M))

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

El siguiente fragmento de código C contiene funciones ODBC intercaladas con funciones SQL de DB2 Spatial Extender que insertan datos en la tabla LOTS. La tabla LOTS se creó con dos columnas: la columna lot_id, que identifica de forma exclusiva cada parcela, y la columna de varios polígonos lot, que contiene la geometría de cada parcela.

```
CREATE TABLE lots (lot_id  integer,
                   lot     db2gse.ST_MultiPolygon);
```

La función ShapeToSQL convierte formas en una geometría de DB2 Spatial Extender. La sentencia INSERT completa se copia en shp_sql. La sentencia INSERT contiene marcadores de parámetro para aceptar los datos LOT_id y los datos lot de forma dinámica.

```
/* Crear la sentencia SQL insert para llenar el id de parcela y
  los varios polígonos de parcela. Los interrogantes son marcadores de parámetro
  que indican los valores de lot_id y lot que se recuperarán en el momento
  de la ejecución. */

strcpy (shp_sql,"insert into lots (lot_id, lot) values(?,
db2gse.ShapeToSQL(cast(? as blob(1m))))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */

rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */

rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Vincular el valor de clave de entero con el primer parámetro. */

pcbvalue1 = 0;

rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Vincular la forma con el segundo parámetro. */
```

```
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* Ejecutar la sentencia insert. */

rc = SQLExecute (hstmt);
```

ST_Area

ST_Area toma un polígono o varios polígonos y devuelve su área.

Sintaxis

db2gse.ST_Area(s db2gse.ST_Surface)

Tipo devuelto

Double

Ejemplos

Un ingeniero municipal necesita una lista de las áreas edificadas. Para obtener la lista, un técnico de GIS selecciona el ID de edificio y el área que ocupa cada edificio.

El área que ocupa cada edificio se almacena en la tabla BUILDINGFOOTPRINTS que se creó con la siguiente sentencia CREATE TABLE:

```
CREATE TABLE BUILDINGFOOTPRINTS ( building_id integer,
                                        lot_id      integer,
                                  footprint   db2gse.ST_MultiPolygon);
```

Para satisfacer la petición del ingeniero municipal, el técnico utiliza la siguiente sentencia SELECT para seleccionar la clave exclusiva, el id de edificio y el área que ocupa cada edificio de la tabla BUILDINGFOOTPRINTS:

```
SELECT building_id, db2gse.ST_Area (footprint) "Area"
FROM BUILDINGFOOTPRINTS;
```

La sentencia SELECT devuelve el siguiente conjunto de resultados:

```
building_id   Area
------------  ------------------------
         506    +1.40768000000000E+003
        1208    +2.55759000000000E+003
         543    +1.80786000000000E+003
         178    +2.08671000000000E+003
         .
         .
         .
```

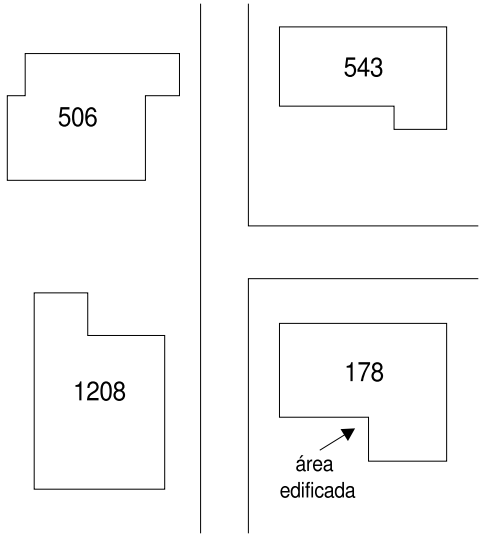


Figura 28. Utilización de área para saber el área edificada. Cuatro de las áreas edificadas y los números de ID de sus correspondientes edificios se muestran junto con la calle adyacente.

ST_AsBinary

ST_AsBinary toma un objeto geometría y devuelve su representación de binario conocido.

Sintaxis

db2gse.ST_AsBinary(g db2gse.ST_Geometry)

Tipo devuelto

BLOB(1m)

Ejemplos

El siguiente fragmento de código ilustra cómo la función ST_AsBinary convierte los varios polígonos de áreas edificadas de la tabla BUILDINGFOOTPRINTS en varios polígonos WKB. Estos varios polígonos se pasan a la función draw_polygon de la aplicación para que los muestre.

```
/* Crear la expresión SQL. */
strcpy(sqlstmt, "select db2gse.ST_AsBinary (footprint) from BUILDINGFOOTPRINTS
where db2gse.EnvelopesIntersect(footprint, db2gse.ST_PolyFromWKB
(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Preparar la sentencia SQL. */
SQLPrepare(hstmt, (UCHAR *)sqlstmt, SQL_NTS);

/* Definir la longitud pcbvalue1 de la forma. */
pcbvalue1 = blob_len;

/* Vincular el parámetro de forma. */
SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, SQL_BLOB, blob_len,
0, shape, blob_len, &pcbvalue1);

/* Ejecutar la consulta. */
rc = SQLExecute (hstmt);

/* Asignar los resultados de la consulta, (los polígonos de zona) a la
  variable fetched_binary.
 */
SQLBindCol (hstmt, 1, SQL_C_Binary, fetched_binary, 100000, &ind_blob);

/* Buscar y cargar cada polígono dentro de la ventana de visualización
y mostrarlo. */
while(SQL_SUCCESS == (rc = SQLFetch(hstmt)))
  draw_polygon(fetched_binary);
```

ST_AsText

db2gse.ST_AsText toma un objeto geometría y devuelve su representación de texto conocido.

Sintaxis

db2gse.ST_AsText(g db2gse.ST_Geometry)

Tipo devuelto

Varchar(4000)

Ejemplos

En el siguiente escenario, la función db2gse.ST_AsText convierte el punto de ubicación HAZARDOUS_SITES en su descripción de texto:

```
CREATE TABLE HAZARDOUS_SITES ( site_id   integer,
                               name      varchar(40),
                               location  db2gse.ST_Point);
INSERT INTO HAZARDOUS_SITES
VALUES (102,
        'W. H. Kleenare Chemical Repository',
        db2gse.ST_PointFromText('point (1020.12 324.02)',
db2gse.coordref()..srid(0)));

SELECT site_id, name, cast(db2gse.ST_AsText(location) as varchar(40))
    "Location"
FROM HAZARDOUS_SITES;
```

La sentencia SELECT devuelve el siguiente conjunto de resultados:

```
SITE_ID Name                                Location
------- ----------------------------------- ---------------------------------
    102 W. H. Kleenare Chemical Repository  POINT (1020.00000000 324.00000000)
```

ST_Boundary

ST_Boundary toma un objeto geometría y devuelve su límite combinado como un objeto geometría.

Sintaxis

db2gse.ST_Boundary(g db2gse.ST_Geometry)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

En el siguiente fragmento de código, se crea una tabla denominada BOUNDARY_TEST. BOUNDARY_TEST tiene dos columnas: GEOTYPE, que se define como varchar, y G1, que se define como la geometría de superclase. Las siguientes sentencias INSERT insertan cada una de las geometrías de subclase. La función ST_Boundary recupera el límite de cada subclase almacenada en la columna de geometría G1. Observe que la dimensión de la geometría resultante es siempre un nivel menor que la geometría de entrada. Los puntos y las geometrías varios puntos siempre dan como resultado una geometría vacía, dimensión 1. Las series lineales y las geometrías varias series lineales devuelven un límite varios puntos, dimensión 0. Un polígono o una geometría varios polígonos devuelve siempre un límite varias series lineales, dimensión 1.

```
CREATE TABLE BOUNDARY_TEST (GEOTYPE varchar(20), G1 db2gse.ST_Geometry)

INSERT INTO BOUNDARY_TEST
VALUES('Point',
       db2gse.ST_PointFromText('point (10.02 20.01)',
                     db2gse.coordref()..srid(0)))

INSERT INTO BOUNDARY_TEST
VALUES('Linestring',
       db2gse.ST_LineFromText('linestring (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO BOUNDARY_TEST
VALUES('Polygon',
       db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,25.02 34.15,
                              19.15 33.94, 10.02 20.01))',
                     db2gse.coordref()..srid(0)))

INSERT INTO BOUNDARY_TEST
VALUES('Multipoint',
       db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO BOUNDARY_TEST
VALUES('Multilinestring',
       db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,10.32 23.98,
11.92 25.64), (9.55 23.75,15.36 30.11))',
```

```
                    db2gse.coordref()..srid(0)))

INSERT INTO BOUNDARY_TEST
VALUES('Multipolygon',
       db2gse.ST_MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,
                                  25.02 34.15, 19.15 33.94,10.02 20.01)),
                                      ((51.71 21.73,73.36 27.04,71.52 32.87,
                                        52.43 31.90,51.71 21.73)))',
                    db2gse.coordref()..srid(0)))

SELECT GEOTYPE,
       CAST(db2gse.ST_AsText(db2gse.ST_Boundary (G1)) as varchar(280))
      "The boundary"
FROM BOUNDARY_TEST


GEOTYPE              The boundary
-------------------- ------------------------------------------------------------
Point                POINT EMPTY
Linestring           MULTIPOINT ( 10.02000000 20.01000000, 11.92000000
 25.64000000)
Polygon              MULTILINESTRING (( 10.02000000 20.01000000, 19.15000000
 33.94000000, 25.02000000 34.15000000, 11.92000000 35.64000000, 10.02000000
 20.01000000))
Multipoint           POINT EMPTY
Multilinestring      MULTIPOINT ( 9.55000000 23.75000000, 10.02000000
 20.01000000, 11.92000000 25.64000000, 15.36000000 30.11000000)
Multipolygon         MULTILINESTRING (( 51.71000000 21.73000000, 73.36000000
 27.04000000, 71.52000000 32.87000000, 52.43000000 31.90000000, 51.71000000
 21.73000000),( 10.02000000 20.01000000, 19.15000000 33.94000000, 25.02000000
 34.15000000, 11.92000000 35.64000000, 10.02000000 20.01000000))

6 record(s) selected.
```

ST_Buffer

ST_Buffer toma un objeto geometría y una distancia y devuelve el objeto geometría que rodea al objeto fuente.

Sintaxis

db2gse.ST_Buffer(g db2gse.ST_Geometry , adistance Double)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

Supongamos que el supervisor de un condado necesita una lista de lugares peligrosos cuyo radio de cinco millas solape áreas especialmente sensibles, como colegios, hospitales y guarderías. Las áreas sensibles se almacenan en la tabla SENSITIVE_AREAS que se crea con la siguiente sentencia CREATE TABLE. La columna ZONE está definida como un polígono, que se almacena como el contorno de cada área sensible.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                               size     float,
                               type     varchar(10),
                               zone     db2gse.ST_Polygon);
```

Los sitios peligrosos se almacenan en la tabla HAZARDOUS_SITES que se crea con la siguiente sentencia CREATE TABLE. La columna LOCATION, definida como un punto, almacena la ubicación que constituye el centro geográfico de cada sitio peligroso.

```
CREATE TABLE HAZARDOUS_SITES ( site_id   integer,
                              name      varchar(128),
                               location  db2gse.ST_Point);
```

Las tablas SENSITIVE_AREAS y HAZARDOUS_SITES se unen mediante la función db2gse.ST_Overlaps. La función devuelve 1 (TRUE) para todas las filas de SENSITIVE_AREAS cuyos polígonos de zona se solapan con el radio de cinco millas que rodea el punto de la ubicación HAZARDOUS_SITES.

```
SELECT sa.name "Sensitive Areas", hs.name "Hazardous Sites"
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
WHERE db2gse.ST_Overlaps(sa.zone, db2gse.ST_Buffer (hs.location,(5 * 5280)))
=  1;
```

En la Figura 29 en la página 200, algunas de las áreas sensibles de este distrito administrativo quedan dentro de las cinco millas que rodean las ubicaciones de los sitios peligrosos. Ambas zonas que rodean con un radio de cinco millas forman intersección con el hospital, y una forma intersección con el colegio. Sin embargo, la guardería queda fuera de ambos radios.

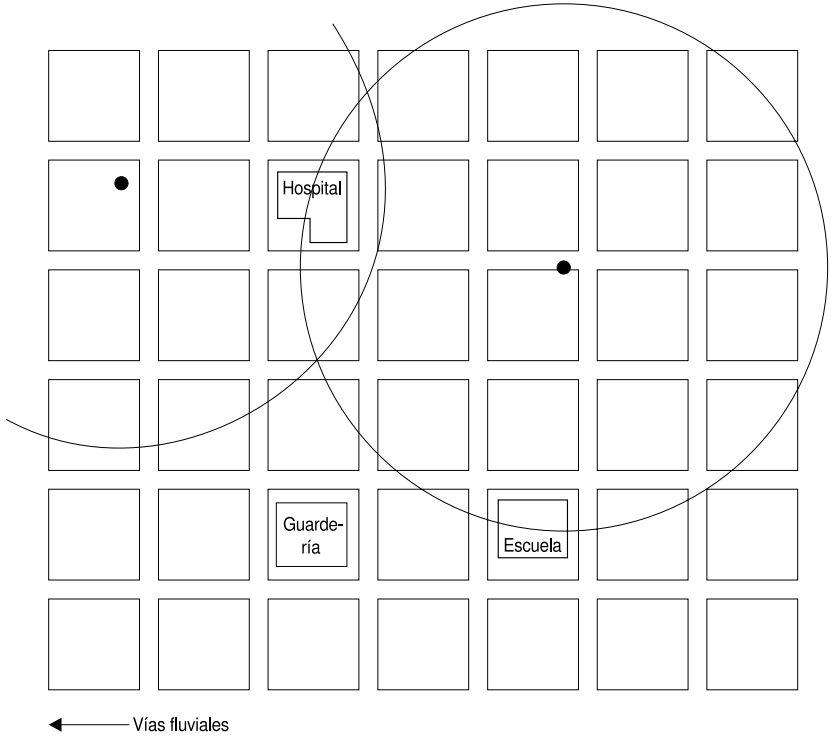


Figura 29. Se redondea un punto con un radio de cinco millas

ST_Centroid

ST_Centroid toma un polígono o varios polígonos y devuelve su centro geométrico como un punto.

Sintaxis

db2gse.ST_Centroid(s db2gse.ST_Surface)
db2gse.ST_Centroid(ms db2gse.ST_MultiSurface)

Tipo devuelto

Para superficie: db2gse.ST_Point

Ejemplos

El técnico municipal de GIS desea visualizar la geometría varios polígonos de las áreas edificadas como puntos en un gráfico de densidad de edificación.

Las áreas edificadas se almacenan en la tabla BUILDINGFOOTPRINTS que se creó con la siguiente sentencia CREATE TABLE.

```
CREATE TABLE BUILDINGFOOTPRINTS (building_id integer,
                                 lot_id      integer,
                                 footprint   db2gse.ST_MultiPolygon);
```

La función ST_Centroid devuelve el centro de cada geometría varios polígonos de áreas edificadas. La función AsBinaryShape convierte el punto central en una forma, la representación externa que reconoce la aplicación.

```
SELECT building_id,
       CAST(db2gse.AsBinaryShape(db2gse.ST_Centroid (footprint)) as blob(1m))
"Centroid"
FROM BUILDINGFOOTPRINTS;
```

ST_Contains

ST_Contains toma dos objetos de geometría y devuelve 1 (TRUE) si el primer objeto contiene el segundo por completo y 0 (FALSE) si no es así.

Sintaxis

db2gse.ST_Contains(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

En el ejemplo siguiente se crean dos tablas. Una tabla contiene las áreas edificadas de la ciudad y la otra contiene sus parcelas. El ingeniero municipal desea asegurarse de que todas las áreas edificadas quedan completamente dentro de sus parcelas.

En ambas tablas el tipo de datos varios polígonos almacena la geometría de las áreas edificadas y sus parcelas. El diseñador de la base de datos ha seleccionado varios polígonos para ambas funciones. El diseñador se ha dado cuenta de que las parcelas pueden estar separadas por funciones naturales, como un río, y que las áreas edificadas pueden constar de varios edificios.

```
CREATE TABLE BUILDINGFOOTPRINTS (building_id integer,
                                 lot_id      integer,
                                 footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );
```

El ingeniero municipal selecciona primero los edificios que no quedan completamente dentro de una parcela.

```
SELECT building_id
  FROM BUILDINGFOOTPRINTS, LOTS
WHERE db2gse.ST_Contains(lot,footprint) = 0;
```

El ingeniero municipal se da cuenta de que la primera consulta devolverá una lista de todos los ID de edificios cuyas áreas quedan fuera de un polígono de parcela. Pero el ingeniero municipal también sabe que esta información no indica si los otros edificios tienen asignado el ID de parcela correcto. Esta segunda consulta realiza una comprobación de integridad de los datos de la columna lot_id de la tabla BUILDINGFOOTPRINTS.

```
SELECT bf.building_id "building id", bf.lot_id "buildings lot_id",
       LOTS.lot_id "LOTS lot_id"
  FROM BUILDINGFOOTPRINTS bf, LOTS
WHERE db2gse.ST_Contains(lot,footprint) = 1 AND LOTS.lot_id <> bf.lot_id;
```

En la Figura 30 en la página 203, las áreas edificadas marcadas con sus ID de edificio quedan dentro de sus parcelas. Las líneas de las parcelas se muestran con líneas de puntos. Aunque no se muestre, estas líneas alcanzan la línea

central de calles y comprenden por completo las parcelas y las áreas edificadas que contienen.

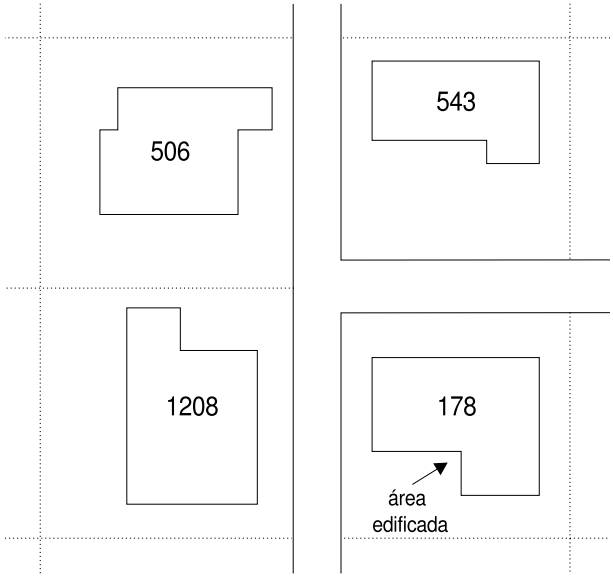


Figura 30. Utilización de ST_Contains para asegurar que todos los edificios quedan dentro de sus parcelas

ST_ConvexHull

ST_ConvexHull toma un objeto geometría y devuelve el casco convexo.

Sintaxis

db2gse.ST_ConvexHull(g db2gse.ST_Geometry)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

El ejemplo crea la tabla CONVEXHULL_TEST que tiene dos columnas: GEOTYPE y G1. La columna GEOTYPE, tipo varchar(20), almacenará el nombre de la subclase de geometría que se almacena en G1, que está definida como una geometría.

```
CREATE TABLE CONVEXHULL_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Cada sentencia INSERT inserta una geometría de cada tipo de subclase en la tabla CONVEXHULL_TEST.

```
INSERT INTO CONVEXHULL_TEST
VALUES('Point',
       db2gse.ST_PointFromText('point (10.02 20.01)',
db2gse.coordref()..srid(0)))

INSERT INTO CONVEXHULL_TEST
VALUES('Linestring',
       db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO CONVEXHULL_TEST
VALUES('Polygon',
       db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,25.02 34.15,
                                         19.15 33.94,10.02 20.01))',
                    db2gse.coordref()..srid(0)))

INSERT INTO CONVEXHULL_TEST
VALUES('Multipoint',
       db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO CONVEXHULL_TEST
VALUES('Multilinestring',
       db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,10.32 23.98,
       11.92 25.64),(9.55 23.75,15.36 30.11))',
                     db2gse.coordref()..srid(0)))

INSERT INTO CONVEXHULL_TEST
VALUES('Multipolygon',
       db2gse.ST_MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,
       25.02 34.15, 19.15 33.94,10.02 20.01)),
```

```
                                ((51.71 21.73,73.36 27.04,71.52 32.87,
                                  52.43 31.90,51.71 21.73)))',
                    db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT lista el nombre de las subclases almacenadas en la columna GEOTYPE y el casco convexo. El casco convexo generado por la función ST_ConvexHull se convierte en texto mediante la función ST_AsText. Luego se convierte a varchar(256) porque la salida por omisión de ST_AsText es varchar(4000).

```
SELECT GEOTYPE, CAST(db2gse.ST_AsText(db2gse.ST_ConvexHull(G1)))
as varchar(256) "The convexhull"
FROM CONVEXHULL_TEST
```

ST_CoordDim

ST_CoordDim devuelve las dimensiones de coordenadas del valor ST_Geometry. Para ver una explicación de las dimensiones de las coordenadas, consulte “Puntos” en la página 138.

Sintaxis

db2gse.ST_CoordDim(g1 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

La tabla coorddim_test se crea con las columnas geotype y g1. La columna geotype almacena el nombre de la subclase de geometría almacenada en la columna de geometría g1.

```
CREATE TABLE coorddim_test (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Las sentencias INSERT insertan una subclase de ejemplo en la tabla coorddim_test.

```
INSERT INTO coorddim_test VALUES(
    'Point', db2gse.ST_PointFromText('point (10.02 20.01)',
db2gse.coordref()..srid(0))
)

INSERT INTO coorddim_test VALUES(
'Linestring',
       db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,11.92 25.64)',
db2gse.coordref()..srid(0))
)

INSERT INTO coorddim_test VALUES(
'Polygon', db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,
25.02 34.15, 19.15 33.94,10.02 20.01))', db2gse.coordref()..srid(0))
)

INSERT INTO coorddim_test VALUES(
'Multipoint', db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0))
)

INSERT INTO coorddim_test VALUES(
'Multilinestring', db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,
10.32 23.98,11.92 25.64),(9.55 23.75,15.36 30.11))', db2gse.coordref()..srid(0))
)

INSERT INTO coorddim_test VALUES(
'Multipolygon',
MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,25.02 34.15,
19.15 33.94,10.02 20.01)),((51.71 21.73,73.36 27.04,71.52 32.87,
52.43 31.90,51.71 21.73)))', db2gse.coordref()..srid(0))
)
```

La sentencia SELECT lista el nombre de subclase almacenado en la columna geotype con la dimensión de coordenadas de dicho tipo de geometría.

```
SELECT geotype, db2gse.ST_coordDim(g1)' coordinate_dimension'
FROM coorddim_test

GEOTYPE              coordinate_dimension
-------------------- -------------------------
ST_Point                                    2
ST_Linestring                               2
ST_Polygon                  2
ST_Multipoint                               2
ST_Multilinestring                          2
ST_Multipolygon             2

6 record(s) selected.
```

ST_Crosses

ST_Crosses toma dos objetos geometría y devuelve 1 (TRUE) si su intersección da como resultado un objeto geometría cuya dimensión es un nivel inferior a la dimensión máxima de los objetos fuente. El objeto intersección contiene puntos interiores a ambas geometrías fuente y no es igual a ninguno de los objetos fuente. Si no es así, devuelve 0 (FALSE).

Sintaxis

db2gse.ST_Crosses(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

El gobierno regional está considerando la posibilidad de sacar una nueva ley según la cual los almacenes de residuos peligrosos de la región no pueden estar a menos de cinco millas de una vía fluvial. El gestor de GIS regional tiene una representación precisa de ríos y corrientes, que se almacenan como varias series lineales en la tabla WATERWAYS. Pero el gestor de GIS sólo tiene una ubicación de un solo punto para cada uno de los almacenes de residuos peligrosos.

```
CREATE TABLE WATERWAYS (id     integer,
                        name       varchar(128),
                        water      db2gse.ST_MultiLineString);

CREATE TABLE HAZARDOUS_SITES ( site_id   integer,
                        name       varchar(128),
                         location  db2gse.ST_Point);
```

Para determinar si el supervisor regional tiene que avisar a los almacenes que están violando la nueva ley propuesta, el gestor de GIS debe rodear las ubicaciones de sitios peligrosos para ver si algún río o corriente cruza el polígono rodeado. El predicado ST_Crosses compara los sitios de la tabla HAZARDOUS_SITES rodeados con WATERWAYS. De este modo sólo se devuelven los registros en los que la vía fluvial cruza el radio de la región sobre el que se aplica la ley propuesta.

```
SELECT ww.name "River or stream", hs.name "Hazardous site"
FROM WATERWAYS ww, HAZARDOUS_SITES hs
WHERE db2gse.ST_Crosses(db2gse.ST_Buffer(hs.location,(5 * 5280)),ww.water)
=  1;
```

En la Figura 31 en la página 209, el área que rodea con un radio de cinco millas los sitios de residuos peligrosos cruza la red de corrientes que fluye por el distrito administrativo de la región. La red de corrientes se ha definido como varias series lineales. De este modo, el conjunto de resultados incluye

todos los segmentos de series lineales que forman parte de los segmentos que cruzan el radio.

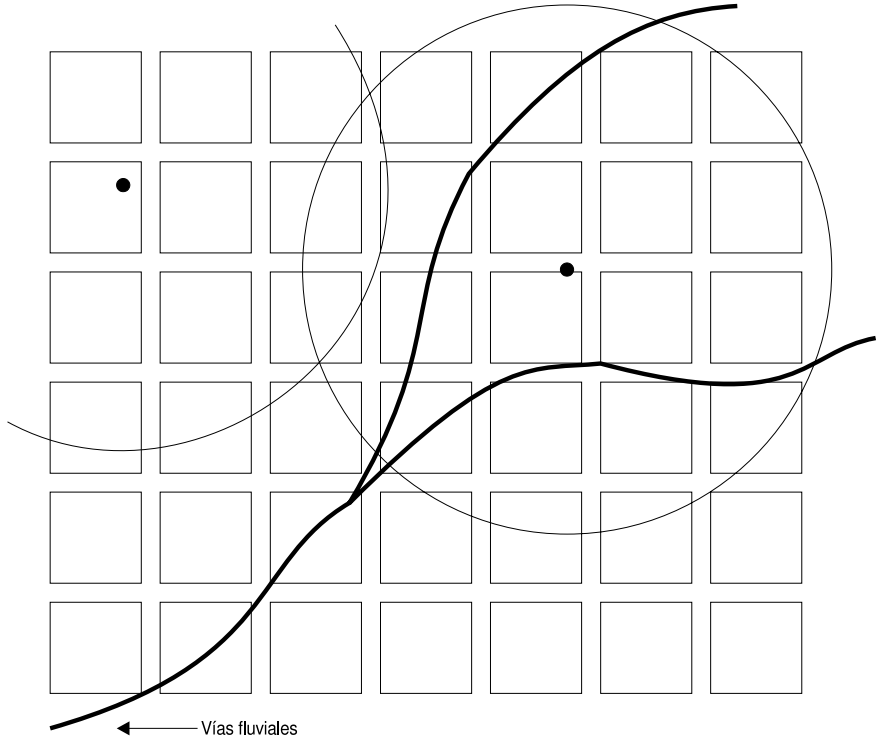


Figura 31. Utilización de ST_Crosses para ver qué vías fluviales pasan por un área de residuos peligrosos

ST_Difference

ST_Difference toma dos objetos geometría y devuelve un objeto geometría que es la diferencia de los objetos fuente.

Sintaxis

db2gse.ST_Difference(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

El ingeniero municipal desea saber el área total de las áreas de parcela de la ciudad que no están edificadas. Es decir, desea saber la suma de las áreas de parcelas sin contar las áreas edificadas.

```
CREATE TABLE BUILDINGFOOTPRINTS (building_id integer,
                                         lot_id      integer,
                                         footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS ( lot_id   integer,
                        lot   db2gse.ST_MultiPolygon);
```

El ingeniero municipal une las tablas BUILDINGFOOTPRINTS y LOTS por la columna lot_id. Luego el ingeniero municipal toma la suma del área de la diferencia de las parcelas menos las áreas construidas.

```
SELECT SUM(db2gse.ST_Area(db2gse.ST_Difference(lot,footprint)))
  FROM BUILDINGFOOTPRINTS bf, LOTS
WHERE bf.lot_id = LOTS.lot_id;
```

ST_Dimension

ST_Dimension toma un objeto geometría y devuelve su dimensión.

Sintaxis

db2gse.ST_Dimension(g1 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

La tabla DIMENSION_TEST se crea con las columnas GEOTYPE y G1. La columna GEOTYPE almacena el nombre de la subclase de geometría que se almacena en la columna de geometría G1.

```
CREATE TABLE DIMENSION_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Las sentencias INSERT insertan una subclase de ejemplo en la tabla DIMENSION_TEST.

```
INSERT INTO DIMENSION_TEST
VALUES('Point',
       db2gse.ST_PointFromText('point (10.02 20.01)',
                                    db2gse.coordref()..srid(0)))

INSERT INTO DIMENSION_TEST
VALUES('Linestring',
       db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,
              11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO DIMENSION_TEST
VALUES('Polygon',
       db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,25.02 34.15,
                                        19.15 33.94,10.02 20.01))',
                     db2gse.coordref()..srid(0)))

INSERT INTO DIMENSION_TEST
VALUES('Multipoint',
       db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,
               11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO DIMENSION_TEST
VALUES('Multilinestring',
       db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,10.32 23.98,
              11.92 25.64),(9.55 23.75,15.36 30.11))',
                      db2gse.coordref()..srid(0)))

INSERT INTO DIMENSION_TEST
VALUES('Multipolygon',
       db2gse.ST_MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,
              25.02 34.15,19.15 33.94,10.02 20.01)),
              ((51.71 21.73,73.36 27.04,71.52 32.87,
              52.43 31.90,51.71 21.73)))',
db2gse.coordref)..srid(0)))
```

La siguiente sentencia SELECT lista el nombre de subclase almacenado en la columna GEOTYPE con la dimensión de este tipo de geometría.

```
SELECT geotype, db2gse.ST_Dimension(g1) "The dimension"
FROM DIMENSION_TEST
```

Se devuelve el siguiente conjunto de resultados.

```
GEOTYPE              The dimension
-------------------- -------------
ST_Point                       0
ST_Linestring                  1
ST_Polygon                     2
ST_Multipoint                  0
ST_Multilinestring             1
ST_Multipolygon                2

6 record(s) selected.
```

ST_Disjoint

ST_Disjoint toma dos geometrías y devuelve 1 (TRUE) si la intersección de las dos geometrías da como resultado un conjunto vacío y 0 (FALSE) si no es así.

Sintaxis

db2gse.ST_Disjoint(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

Una compañía de seguros desea evaluar la cobertura de seguro del hospital, las guarderías y los colegios de una ciudad. Parte de este proceso incluye determinar la amenaza que suponen los sitios de residuos peligrosos para cada institución. La compañía de seguros desea tener en cuenta las instituciones que no están expuestas a la contaminación. El consultor de GIS contratado por la compañía de seguros debe localizar todas las instituciones que no se encuentran en un radio de cinco millas de un sitio de residuos peligrosos.

La tabla SENSITIVE_AREAS contiene varias columnas que describen las instituciones amenazadas además de la columna ZONE, que almacena la geometría de polígono de la institución.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                               size     float,
                               type     varchar(10),
                               zone     db2gse.ST_Polygon);
```

La tabla HAZARDOUS_SITES almacena la identidad de los sitios en las columnas SITE_ID y NAME, mientras que la ubicación geográfica real de cada sitio se almacena en la columna LOCATION.

```
CREATE TABLE HAZARDOUS_SITES ( site_id   integer,
                              name      varchar(128),
                               location  db2gse.ST_Point);
```

La siguiente sentencia SELECT lista los nombres de todas las áreas sensibles que no se encuentran en un radio de 5 millas de un sitio de residuos peligrosos. La función ST_Intersects podría sustituir a la función ST_Disjoint en esta consulta si el resultado de la función se define como igual a 0 en lugar de 1. Esto se debe a que ST_Intersects y ST_Disjoint devuelven resultados exactamente opuestos.

```
SELECT sa.name
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
WHERE db2gse.ST_Disjoint(db2gse.ST_Buffer(hs.location,(5 * 5280)),sa.zone) = 1;
```

En la Figura 32, se comparan las áreas sensibles con el radio de cinco millas de los sitios de residuos peligrosos. La guardería es la única área sensible en la que la función ST_Disjoint devuelve 1 (TRUE). La función ST_Disjoint devuelve 1 cuando dos geometrías no forman ninguna intersección.

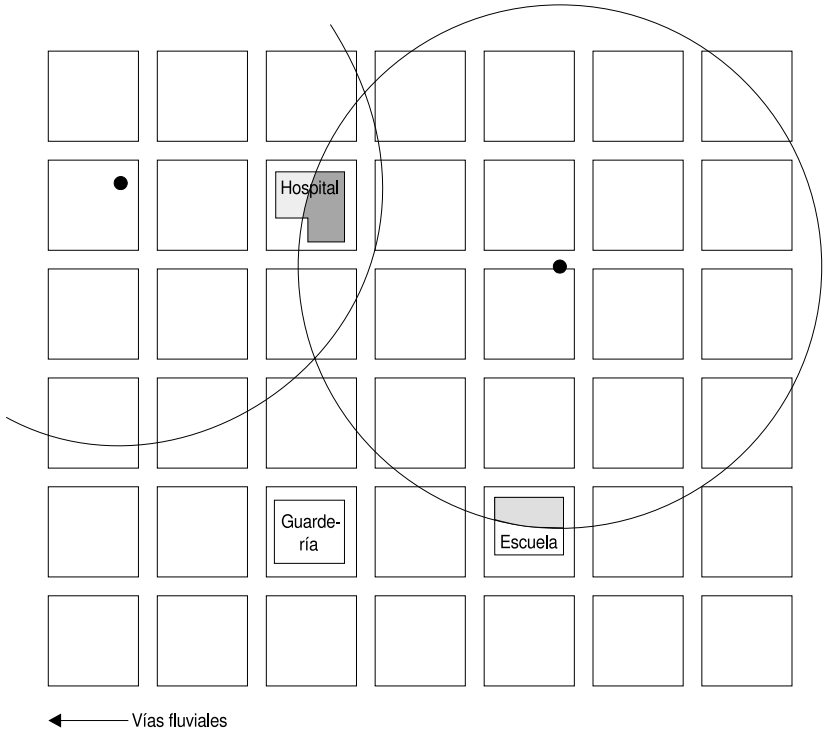


Figura 32. Utilización de ST_Disjoint para buscar edificios que no quedan dentro de ningún área de residuos peligrosos (no forman intersección con la misma)

ST_Distance

ST_Distance toma dos geometrías y devuelve la distancia mínima que las separa.

Sintaxis

db2gse.ST_Distance(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

Double

Ejemplos

El ingeniero municipal necesita una lista de todos los edificios que están a un pie o menos de una línea de parcela.

La columna BUILDING_ID de la tabla BUILDINGFOOTPRINTS identifica de forma exclusiva cada edificio. La columna LOT_ID identifica la parcela a la que pertenece cada edificio. La geometría varios polígonos de áreas edificadas almacena el área edificada correspondiente a cada edificio.

```
CREATE TABLE BUILDINGFOOTPRINTS (   building_id integer,
                                    lot_id      integer,
                                    footprint   db2gse.ST_MultiPolygon);
```

La tabla LOTS almacena el ID de parcela que identifica de forma exclusiva cada parcela y la geometría varios polígonos de parcela que contiene la geometría de líneas de la parcela.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot      db2gse.ST_MultiPolygon);
```

La consulta devuelve una lista de los ID de edificios que están a un pie o menos de sus líneas de parcela. La función ST_Distance realiza una unión espacial entre las geometrías varios polígonos correspondientes a áreas edificadas y al límite de la parcela. Sin embargo, la unión entre bf.lot_id y LOTS.lot_id asegura que la función ST_Distance sólo compara las geometrías varios polígonos que pertenecen a la misma parcela.

```
SELECT bf.building_id
  FROM BUILDINGFOOTPRINTS bf, LOTS
 WHERE bf.lot_id = LOTS.lot_id AND
       db2gse.ST_Distance(bf.footprint, db2gse.ST_Boundary(LOTS.lot)) <= 1.0;
```

ST_Endpoint

ST_Endpoint toma una serie lineal y devuelve un punto que es el último punto de la serie lineal.

Sintaxis

db2gse.ST_Endpoint(c db2gse.ST_Curve)

Tipo devuelto

db2gse.ST_Point

Ejemplos

La tabla ENDPOINT_TEST almacena la columna de enteros GID que identifica de forma exclusiva cada fila y la columna de series lineales LN1 que almacena series lineales.

```
CREATE TABLE ENDPOINT_TEST (gid integer, ln1 db2gse.ST_LineString)
```

Las sentencias INSERT insertan series lineales en la tabla ENDPOINT_TEST. La primera no tiene coordenadas Z ni medidas; la segunda, sí.

```
INSERT INTO ENDPOINT_TEST
VALUES( 1,
        db2gse.ST_LineFromText('linestring (10.02 20.01,23.73 21.92,
        30.10 40.23)', db2gse.coordref()..srid(0)))

INSERT INTO ENDPOINT_TEST
VALUES (2,
       db2gse.ST_LineFromText('linestring zm (10.02 20.01 5.0 7.0,
       23.73 21.92 6.5 7.1, 30.10 40.23 6.9 7.2)',
                    db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT lista la columna GID con la salida de la función ST_Endpoint. La función ST_Endpoint genera una geometría punto que la función ST_AsText convierte en texto. Se utiliza la función CAST para abreviar el valor varchar(4000) por omisión de la función ST_AsText a un valor varchar(60).

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.ST_Endpoint(ln1)) AS varchar(60))
"Endpoint"
FROM ENDPOINT_TEST
```

Se devuelve el siguiente conjunto de resultados.

```
GID         Endpoint
----------- ------------------------------------------------------------
          1 POINT ( 30.10000000 40.23000000)
          2 POINT ZM ( 30.10000000 40.23000000 7.00000000 7.20000000)

  2 record(s) selected.
```

ST_Envelope

ST_Envelope toma un objeto geometría y devuelve el recuadro que lo limita como una geometría.

Sintaxis

db2gse.ST_Envelope(g db2gse.ST_Geometry)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

La columna GEOTYPE de la tabla ENVELOPE_TEST almacena el nombre de la subclase de geometría almacenada en la columna de geometría G1.

```
CREATE TABLE ENVELOPE_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Las siguientes sentencias INSERT insertan cada subclase de geometría en la tabla ENVELOPE_TEST.

```
INSERT INTO ENVELOPE_TEST
VALUES('Point',
       db2gse.ST_PointFromText('point (10.02 20.01)',
                                    db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES('Linestring',
        db2gse.ST_LineFromText('linestring (10.01 20.01, 10.01 30.01,
        10.01 40.01)', db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES('Linestring',
       db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,11.92 25.64)',
       db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES('Polygon',
       db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,25.02 34.15,
                                        19.15 33.94,10.02 20.01))',
                     db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES('Multipoint',
       db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,11.92 25.64)',
       db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES('Multilinestring',
       db2gse.ST_MLineFromText('multilinestring ((10.01 20.01,20.01 20.01,
       30.01 20.01), (30.01 20.01,40.01 20.01,50.01 20.01))',
                      db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES('Multilinestring',
```

```
       db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,10.32 23.98,
       11.92 25.64), ( 9.55 23.75,15.36 30.11))',
                     db2gse.coordref()..srid(0)))

INSERT INTO ENVELOPE_TEST
VALUES('Multipolygon',
       db2gse.ST_MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,
       25.02 34.15, 19.15 33.94,10.02 20.01)),
                                       ((51.71 21.73,73.36 27.04,71.52 32.87,
                                         52.43 31.90,51.71 21.73)))',
                     db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT lista el nombre de subclase junto a su envoltura. Puesto que la función ST_Envelope devuelve un punto, una serie lineal o un polígono, su salida se convierte en texto con la función ST_AsText. La función CAST convierte el resultado varchar(4000) por omisión de la función ST_AsText a un valor varchar(280).

```
SELECT GEOTYPE, CAST(db2gse.ST_AsText(db2gse.ST_Envelope(g1)) AS varchar(280))
"The envelope"
FROM ENVELOPE_TEST
```

Se devuelve el siguiente conjunto de resultados.

```
GEOTYPE              The envelope
-------------------- ------------------------------------------------------------
Point                POINT ( 10.02000000 20.01000000)
Linestring           LINESTRING ( 10.01000000 20.01000000, 10.01000000
 40.01000000)
Linestring           POLYGON (( 10.02000000 20.01000000, 11.92000000
 20.01000000, 11.92000000 25.64000000, 10.02000000 25.64000000, 10.02000000
 20.01000000))
Polygon              POLYGON (( 10.02000000 20.01000000, 25.02000000
 20.01000000, 25.02000000 35.64000000, 10.02000000 35.64000000, 10.02000000
 20.01000000))
Multipoint           POLYGON (( 10.02000000 20.01000000, 11.92000000
 20.01000000, 11.92000000 25.64000000, 10.02000000 25.64000000, 10.02000000
 20.01000000))
Multilinestring      LINESTRING ( 10.01000000 20.01000000, 50.01000000
 20.01000000)
Multilinestring      POLYGON (( 9.55000000 20.01000000, 15.36000000
 20.01000000, 15.36000000 30.11000000, 9.55000000 30.11000000, 9.55000000
 20.01000000))
Multipolygon         POLYGON (( 10.02000000 20.01000000, 73.36000000
 20.01000000, 73.36000000 35.64000000, 10.02000000 35.64000000, 10.02000000
 20.01000000))

  8 record(s) selected.
```

ST_Equals

ST_Equals compara dos geometrías y devuelve 1 (TRUE) si las geometrías son idénticas y 0 (FALSE) si no es así.

Sintaxis

db2gse.ST_Equals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

El técnico de GIS sospecha que algunos datos de la tabla BUILDINGFOOTPRINTS están duplicados. El técnico consulta la tabla para determinar si algunas de las geometrías varios polígonos de las áreas edificadas son iguales.

La tabla BUILDINGFOOTPRINTS se crea con la siguiente sentencia. La columna BUILDING_ID identifica de forma exclusiva los edificios, la columna LOT_ID identifica la parcela del edificio y la columna FOOTPRINT almacena la geometría del edificio.

```
CREATE TABLE BUILDINGFOOTPRINTS ( building_id integer,
                                  lot_id      integer,
                                  footprint   db2gse.ST_MultiPolygon);
```

La tabla BUILDINGFOOTPRINTS se une espacialmente a sí misma mediante el predicado ST_Equals, el cual devuelve 1 si encuentra dos geometrías varios polígonos iguales. Se necesita la condición bf1.building_id <> bf2.building_id para eliminar la comparación de la misma geometría.

```
SELECT bf1.building_id, bf2.building_id
FROM BUILDINGFOOTPRINTS bf1, BUILDINGFOOTPRINTS bf2
WHERE db2gse.ST_Equals(bf1.footprint,bf2.footprint) = 1
      and bf1.building_id <> bf2.building_id;
```

ST_ExteriorRing

ST_ExteriorRing toma un polígono y devuelve su anillo exterior como una serie lineal.

Sintaxis

db2gse.ST_ExteriorRing(s db2gse.ST_Polygon)

Tipo devuelto

db2gse.ST_LineString

Ejemplos

Un ornitólogo que está estudiando la población de aves de varias islas de los mares del sur sabe que la zona en la que se alimenta una determinada especie de aves está restringida a la orilla. Para calcular la capacidad de las islas para alimentar a esta especie, el ornitólogo necesita saber el perímetro de las islas. Aunque algunas de las islas contienen estanques, las orillas de estos estanques están habitadas exclusivamente por otras especies de aves más agresivas. Por lo tanto, el ornitólogo necesita saber el perímetro del anillo exterior de las islas.

Las columnas ID y NAME de la tabla ISLANDS identifican cada isla y la columna LAND de tipo ST_Polygon almacena la geometría de cada una de ellas.

```
CREATE TABLE ISLANDS (id   integer,
                      name   varchar(32),
                    land  db2gse.ST_Polygon);
```

La función ST_ExteriorRing extrae el anillo exterior del polígono de cada isla como una serie lineal. La longitud de la serie lineal se establece mediante la función length. Las longitudes de las series lineales se suman mediante la función SUM.

```
SELECT SUM(db2gse.ST_length(db2gse.ST_ExteriorRing (land))) FROM ISLANDS;
```

En la Figura 33 en la página 221, los anillos exteriores de las islas representan la interfaz ecológica que cada isla comparte con el mar. Algunas de las islas tienen lagos, que se representan mediante los anillos interiores de los polígonos.

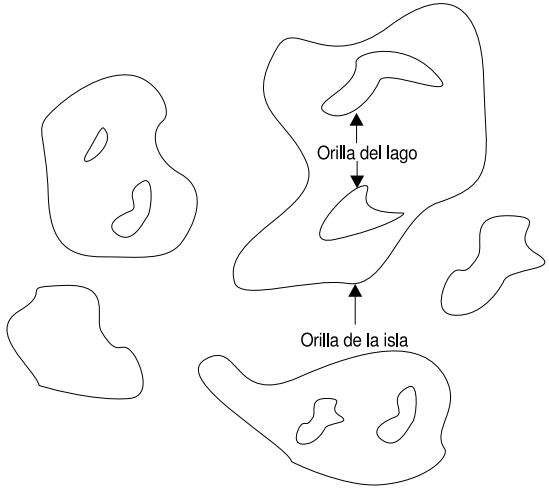


Figura 33. Utilización de ST_ExteriorRing para determinar la longitud de la orilla de una isla

ST_GeometryFromText

ST_GeometryFromText toma una representación de texto conocido y una identidad del sistema de referencias espaciales y devuelve un objeto geometría.

Sintaxis

db2gse.ST_GeometryFromText(geometryTaggedText Varchar(4000), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

La tabla GEOMETRY_TEST contiene la columna de enteros GID, que identifica de forma exclusiva cada fila, y la columna G1, que almacena la geometría.

```
CREATE TABLE GEOMETRY_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Las sentencias INSERT insertan los datos en las columnas GID y G1 de la tabla GEOMETRY_TEST. La función ST_GeometryFromText convierte la representación de texto de cada geometría en su correspondiente subclase de DB2 Spatial Extender de la que se pueden crear instancias.

```
INSERT INTO GEOMETRY_TEST
VALUES(1, db2gse.ST_GeometryFromText('point (10.02 20.01)',
     db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRY_TEST
VALUES (2,
        db2gse.ST_GeometryFromText('linestring (10.01 20.01, 10.01 30.01,
        10.01 40.01)', db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRY_TEST
VALUES(3,
       db2gse.ST_GeometryFromText('polygon ((10.02 20.01,11.92 35.64,25.02 34.15,
                                        19.15 33.94,10.02 20.01))',
                        db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRY_TEST
VALUES(4,
       db2gse.ST_GeometryFromText('multipoint (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRY_TEST
VALUES(5,
       db2gse.ST_GeometryFromText('multilinestring ((10.02 20.01,10.32 23.98,
                                          11.92 25.64),
                                      (9.55 23.75,15.36 30.11))',
                        db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRY_TEST
```

```
VALUES(6,
       db2gse.ST_GeometryFromText('multipolygon (((10.02 20.01,11.92 35.64,
      25.02 34.15, 19.15 33.94,10.02 20.01)),
                                        ((51.71 21.73,73.36 27.04,71.52 32.87,
                                          52.43 31.90,51.71 21.73)))',
                         db2gse.coordref()..srid(0)))
```

ST_GeomFromWKB

ST_GeomFromWKB toma una representación de binario conocido y una identidad del sistema de referencias espaciales y devuelve un objeto geometría.

Sintaxis

db2gse.ST_GeomFromWKB(WKBGeometry Blob(1M), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

El siguiente fragmento de código C contiene funciones ODBC intercaladas con funciones SQL de DB2 Spatial Extender que insertan datos en la tabla LOTS.

La tabla LOTS se creó con dos columnas: la columna LOT_ID, que identifica de forma exclusiva cada parcela, y la columna de varios polígonos LOT, que contiene la geometría de cada parcela.

```
CREATE TABLE LOTS ( lot_id   integer,
                    lot      db2gse.ST_MultiPolygon);
```

La función ST_GeomFromWKB convierte las representaciones WKB en geometría de DB2 Spatial Extender. La sentencia INSERT completa se copia en una serie de caracteres wkb_sql. La sentencia INSERT contiene marcadores de parámetro para aceptar los datos LOT_ID y los datos LOT de forma dinámica.

```
/* Crear la sentencia SQL insert para llenar el id de parcela y
  los varios polígonos de parcela. Los interrogantes son marcadores de parámetro
  que indican los valores de lot_id y lot que se recuperarán en el momento
  de la ejecución. */
strcpy (wkb_sql,"insert into LOTS (lot_id, lot) values (?,
  db2gse.ST_GeomFromWKB

(cast(? as blob(1m)), db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Vincular el valor de clave de entero con el primer parámetro. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Vincular la forma con el segundo parámetro. */
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

ST_GeometryN

ST_GeometryN toma un grupo y un índice de entero y devuelve el objeto de geometría número *n* del grupo.

Sintaxis

db2gse.ST GeometryN(g db2gse.ST_GeomCollection, n Integer)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

El ingeniero municipal desea saber si todas las áreas edificadas quedan dentro del primer polígono de la geometría varios polígonos de la parcela.

La columna BUILDING_ID identifica de forma exclusiva cada fila de la tabla BUILDINGFOOTPRINTS. La columna LOT_ID identifica la parcela del edificio. La columna FOOTPRINT almacena la geometría del edificio.

```
CREATE TABLE BUILDINGFOOTPRINTS ( building_id integer,
                                        lot_id      integer,
                                        footprint   db2gse.ST_MultiPolygon);

CREATE TABLE LOTS ( lot_id   integer,
                    lot   db2gse.ST_MultiPolygon);
```

La consulta lista las columnas building_id y lot_id de la tabla BUILDINGFOOTPRINTS correspondientes a las áreas edificadas que quedan dentro del primer polígono de parcela. La función ST_GeometryN devuelve un primer polígono de parcela en la matriz de varios polígonos.

```
SELECT bf.building_id,bf.lot_id
FROM BUILDINGFOOTPRINTS bf,LOTS
WHERE db2gse.ST_Within(footprint, db2gse.ST_GeometryN (lot,1)) = 1
      AND bf.lot_id = LOTS.lot_id;
```

ST_GeometryType

ST_GeometryType toma un objeto ST_Geometry y devuelve su tipo de geometría como una serie.

Sintaxis

db2gse.ST_GeometryType (g db2gse.ST_Geometry)

Tipo devuelto

Varchar(4000)

Ejemplos

La tabla GEOMETRYTYPE_TEST contiene la columna de geometría G1.

```
CREATE TABLE GEOMETRYTYPE_TEST(g1 db2gse.ST_Geometry)
```

Las siguientes sentencias INSERT insertan cada subclase de geometría en la columna G1.

```
INSERT INTO GEOMETRYTYPE_TEST
VALUES(db2gse.ST_GeometryFromText('point (10.02 20.01)',
       db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRYTYPE_TEST
VALUES (db2gse.ST_GeometryFromText('linestring (10.01 20.01, 10.01 30.01,
        10.01 40.01)', db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRYTYPE_TEST
VALUES(db2gse.ST_Geometrytype_test values(db2gse.ST_GeomFromText('polygon
      ((10.02 20.01,11.92 35.64,25.02 34.15,19.15 33.94, 10.02 20.01))',
                        db2gse.coordref()..srid(0))))

INSERT INTO GEOMETRYTYPE_TEST
VALUES(db2gse.ST_GeometryFromText('multipoint (10.02
 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRYTYPE_TEST
VALUES(db2gse.ST_GeometryFromText('multilinestring ((10.02 20.01,10.32 23.98,
                                          11.92 25.64),
                                      (9.55 23.75,15.36 30.11))',
                        db2gse.coordref()..srid(0)))

INSERT INTO GEOMETRYTYPE_TEST
VALUES(db2gse.ST_GeometryFromText('multipolygon (((10.02 20.01,11.92 35.64,
        25.02 34.15, 19.15 33.94,10.02 20.01)),
                                      ((51.71 21.73,73.36 27.04,71.52 32.87,
                                        52.43 31.90,51.71 21.73)))',
                        db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT lista el tipo de geometría de cada subclase que se almacena en la columna de geometría G1.

```
SELECT db2gse.ST_GeometryType(g1) "Geometry type" FROM GEOMETRYTYPE_TEST
```

Se devuelve el siguiente conjunto de resultados.

```
Geometry type
------------------------------
ST_Point
ST_LineString
ST_Polygon
ST_MultiPoint
ST_MultiLineString
ST_MultiPolygon

6 record(s) selected.
```

ST_InteriorRingN

Devuelve el anillo interior número *n* de un polígono como una serie lineal. Los anillos no se organizan por orientación geométrica. Se organizan según las reglas definidas por las rutinas internas de verificación de geometrías. Así pues, el orden de los anillos no se puede predefinir.

Sintaxis

ST_InteriorRingN(p ST_Polygon, n Integer)

Tipo devuelto

db2gse.ST_LineString

Ejemplos

Un ornitólogo que está estudiando la población de aves de varias islas de los mares del sur sabe que la zona en la que se alimenta una determinada especie pasiva de aves está restringida a la orilla. Algunas de las islas contienen varios lagos. Las orillas de los lagos están habilitadas exclusivamente por especies de aves más agresivas. El ornitólogo sabe que, por cada isla, si el perímetro de los lagos supera un determinado umbral, las especies agresivas serán tan numerosas que amenazarán a las especies pasivas de la orilla. Por lo tanto, el ornitólogo necesita saber la suma de perímetros de los anillos interiores de las islas.

En la Figura 34 en la página 230, los anillos exteriores de las islas representan la interfaz ecológica que cada isla comparte con el mar. Algunas de las islas tienen lagos, que se representan mediante los anillos interiores de los polígonos.

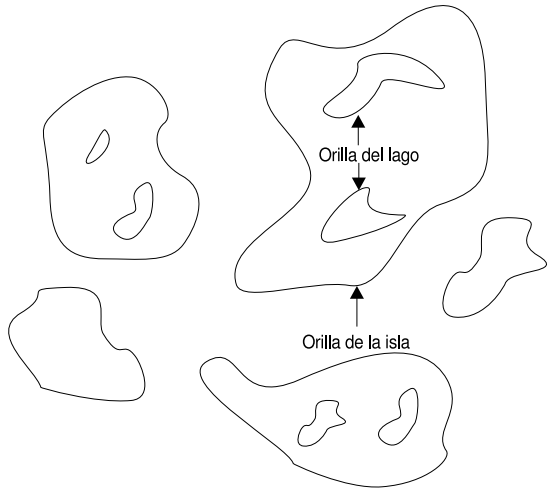


Figura 34. Utilización de ST_InteriorRingN para determinar la longitud de las orillas dentro de cada isla

Las columnas ID y name de la tabla ISLANDS identifican cada isla, mientras que la columna de polígonos land almacena la geometría de la isla.

```
CREATE TABLE ISLANDS (id   integer,
                      name   varchar(32),
                    land  db2gse.ST_Polygon);
```

El siguiente programa ODBC utiliza la función ST_InteriorRingN para extraer el anillo interior (lago) de cada polígono de islas como una serie lineal. El perímetro de la serie lineal que devuelve la función length se suma y se muestra junto con el ID de la isla.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#include "sg.h"
#include "sgerr.h"
#include "sqlcli1.h"

/***                        ***
 *** Cambiar estas constantes ***
 ***                        ***/

#define USER_NAME   "sdetest"   /* su nombre de usuario */
#define USER_PASS   "acid.rain" /* su contraseña de usuario */
#define DB_NAME     "mydb"      /* base de datos a la que se va a conectar */

static void check_sql_err (SQLHDBC  handle,
                           SQLHSTMT hstmt,
```

```
  LONG          rc,
                            CHAR     *str);

void main( argc, argv )
int argc;
char *argv[];
{
  SQLHDBC       handle;
  SQLHENV       henv;
  CHAR          sql_stmt[256];
  LONG          rc,
                total_perimeter,
                num_lakes,
                lake_number,
                island_id,
                lake_perimeter;
  SQLHSTMT      island_cursor,
                lake_cursor;
  SDWORD        pcbvalue,
                id_ind,
                lake_ind,
                length_ind;

/* Asignar memoria para el manejador de entorno ODBC henv e inicializar
la aplicación. */

  rc = SQLAllocEnv (&henv);
    if (rc != SQL_SUCCESS)
  {
    printf ("SQLAllocEnv failed with %d\n", rc);
    exit(0);
  }

/* Asignar memoria para un manejador de conexiones del entorno henv. */

  rc = SQLAllocConnect (henv, &handle);
    if (rc != SQL_SUCCESS)
  {
    printf ("SQLAllocConnect failed with %d\n", rc);
    exit(0);
  }

/* Cargar el controlador ODBC y conectar con la fuente de datos identificada
   por la base de datos, usuario y contraseña. */

  rc = SQLConnect (handle,
                   (UCHAR *)DB_NAME,
                   SQL_NTS,
                   (UCHAR *)USER_NAME,
                   SQL_NTS,
                   (UCHAR *)USER_PASS,
                   SQL_NTS);

  check_sql_err (handle, NULL, rc, "SQLConnect");
```

```
/* Asignar memoria al manejador de sentencia SQL island_cursor. */

  rc = SQLAllocStmt (handle, &island_cursor);
  check_sql_err (handle, NULL, rc, "SQLAllocStmt");

/* Preparar y ejecutar la consulta para obtener los ID de isla y el
   número de lagos (anillos interiores) */

  strcpy (sql_stmt, "select id, db2gse.ST_NumInteriorRings(land) from ISLANDS");

  rc = SQLExecDirect (island_cursor, (UCHAR *)sql_stmt, SQL_NTS);
  check_sql_err (NULL, island_cursor, rc, "SQLExecDirect");

/* Vincular la columna ID de la tabla con la variable island_id */

  rc = SQLBindCol (island_cursor, 1, SQL_C_SLONG, &island_id, 0, &id_ind);
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

/* Vincular el resultado de numinteriorrings(land) con la variable num_lakes. */

  rc = SQLBindCol (island_cursor, 2, SQL_C_SLONG, &num_lakes, 0, &lake_ind);
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

/* Asignar memoria al manejador de sentencia SQL lake_cursor. */

rc = SQLAllocStmt (handle, &lake_cursor);
  check_sql_err (handle, NULL, rc, "SQLAllocStmt");

/* Preparar la consulta para obtener la longitud de un anillo interior. */

  strcpy (sql_stmt,
          "select Length(db2gse.ST_InteriorRingN(land, cast (? as
integer))) from ISLANDS where id = ?");

  rc = SQLPrepare (lake_cursor, (UCHAR *)sql_stmt, SQL_NTS);
  check_sql_err (NULL, lake_cursor, rc, "SQLPrepare");

/* Vincular la variable lake_number como el primer parámetro de entrada */

  pcbvalue = 0;
  rc = SQLBindParameter (lake_cursor, 1, SQL_PARAM_INPUT, SQL_C_LONG,
       SQL_INTEGER, 0, 0, &lake_number, 0, &pcbvalue);
  check_sql_err (NULL, lake_cursor, rc, "SQLBindParameter");

/* Vincular island_id como el segundo parámetro de entrada */

  pcbvalue = 0;
  rc = SQLBindParameter (lake_cursor, 2, SQL_PARAM_INPUT, SQL_C_LONG,
       SQL_INTEGER, 0, 0, &island_id, 0, &pcbvalue);
  check_sql_err (NULL, lake_cursor, rc, "SQLBindParameter");

/* Vincular el resultado de Length(db2gse.ST_InteriorRingN(land, cast
  (? as integer))) a la variable lake_perimeter */

  rc = SQLBindCol (lake_cursor, 1, SQL_C_SLONG, &lake_perimeter, 0,
```

```
                    &length_ind);
  check_sql_err (NULL, island_cursor, rc, "SQLBindCol");

  /* Bucle externo, obtener id de islas y el número de lagos
  (anillos interiores) */

  while (SQL_SUCCESS == rc)
  {
    /* Buscar y cargar una isla */

    rc = SQLFetch (island_cursor);

    if (rc != SQL_NO_DATA)
    {
      check_sql_err (NULL, island_cursor, rc, "SQLFetch");

      /* Bucle interno, para esta isla obtener el perímetro de todos
         sus lagos (anillos interiores) */

      for (total_perimeter = 0,lake_number = 1;
           lake_number <= num_lakes;
           lake_number++)
      {
        rc = SQLExecute (lake_cursor);
        check_sql_err (NULL, lake_cursor, rc, "SQLExecute");

        rc = SQLFetch (lake_cursor);
        check_sql_err (NULL, lake_cursor, rc, "SQLFetch");

        total_perimeter += lake_perimeter;

        SQLFreeStmt (lake_cursor, SQL_CLOSE);
      }
    }

/* Mostrar el id de isla y el perímetro total de sus lagos. */

    printf ("Island ID = %d,  Total lake perimeter = %d\n",
             island_id,total_perimeter);

  }

  SQLFreeStmt (lake_cursor, SQL_DROP);
  SQLFreeStmt (island_cursor, SQL_DROP);
        SQLDisconnect (handle);
        SQLFreeConnect (handle);
  SQLFreeEnv (henv);

  printf( "\nTest Complete ...\n" );

}

static void check_sql_err (SQLHDBC handle, SQLHSTMT  hstmt, LONG rc,
                           CHAR *str)
{
```

```
    SDWORD dbms_err = 0;
    SWORD  length;
    UCHAR   err_msg[SQL_MAX_MESSAGE_LENGTH], state[6];

    if (rc != SQL_SUCCESS)
    {
      SQLError (SQL_NULL_HENV, handle, hstmt, state, &dbms_err,
                err_msg, SQL_MAX_MESSAGE_LENGTH - 1, &length);
      printf ("%s ERROR (%d): DBMS code:%d, SQL state: %s, message:
               \n %s\n", str, rc, dbms_err, state, err_msg);

      if (handle)
      {
        SQLDisconnect (handle);
        SQLFreeConnect (handle);
      }
      exit(1);
    }
}
```

ST_Intersection

ST_Intersection toma dos objetos geometría y devuelve el conjunto de intersecciones como un objeto geometría.

Sintaxis

db2gse.ST_Intersection(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

El jefe de bomberos debe obtener las áreas de los hospitales, colegios y guarderías que forman intersección con el radio de una posible área de contaminación por residuos peligrosos.

Las áreas sensibles se almacenan en la tabla SENSITIVE_AREAS que se crea con la siguiente sentencia CREATE TABLE. La columna ZONE está definida como un polígono que almacena el contorno de cada una de las áreas sensibles.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                               size     float,
                               type     varchar(10),
                               zone     db2gse.ST_Polygon);
```

Los sitios peligrosos se almacenan en la tabla HAZARDOUS_SITES que se crea con la siguiente sentencia CREATE TABLE. La columna LOCATION, definida como un punto, almacena la ubicación que constituye el centro geográfico de cada sitio peligroso.

```
CREATE TABLE HAZARDOUS_SITES ( site_id   integer,
                              name       varchar(128),
                               location  db2gse.ST_Point);
```

La función buffer genera un área con un radio de cinco millas que rodea las ubicaciones de sitios de residuos peligrosos. La función ST_Intersection genera polígonos a partir de la intersección de los polígonos de sitios de residuos peligrosos rodeados y las áreas sensibles. La función ST_Area devuelve el área de los polígonos de intersección, que se suma para cada sitio peligroso mediante la función SUM. La cláusula GROUP BY indica a la consulta que sume las áreas de la intersección por id del sitio de residuos peligrosos.

```
SELECT hs.name,SUM(db2gse.ST_Area(db2gse.ST_Intersection (sa.zone,
db2gse.ST_buffer hs.location,(5 * 5280))))
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
GROUP BY hs.site_id;
```

En la Figura 35 en la página 236, los círculos representan los polígonos rodeados que hay alrededor de los sitios de residuos peligrosos. La

intersección de estos polígonos rodeados con los polígonos de áreas sensibles generan otros tres polígonos. El hospital de la parte superior izquierda forma intersección dos veces, mientras que el colegio de la parte inferior derecha forma intersección una sola vez.

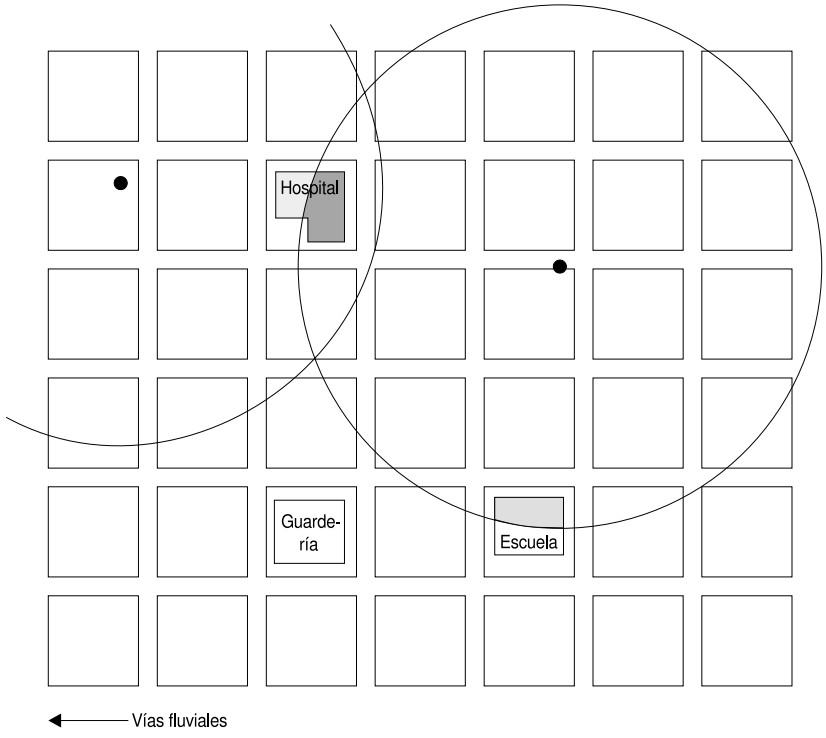


Figura 35. Utilización de ST_Intersection para determinar en qué medida un área de cada uno de los edificios puede verse afectada por los residuos peligrosos

ST_Intersects

ST_Intersects toma dos geometrías y devuelve 1 (TRUE) si la intersección de las dos geometrías no da como resultado un conjunto vació. Si no es así, devuelve 0 (FALSE).

Sintaxis

db2gse.ST_Intersects(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

El jefe de bomberos necesita una lista de todas las áreas sensibles que quedan dentro de un radio de cinco millas de un sitio de residuos peligrosos.

Las áreas sensibles se almacenan en la tabla SENSITIVE_AREAS que se crea con la siguiente sentencia CREATE TABLE. La columna ZONE está definida como un polígono que almacena el contorno de cada una de las áreas sensibles.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                               size     float,
                               type     varchar(10),
                               zone     db2gse.ST_Polygon);
```

Los sitios peligrosos se almacenan en la tabla HAZARDOUS_SITES, creada con la siguiente sentencia CREATE TABLE. La columna LOCATION, definida como un punto, almacena la ubicación que constituye el centro geográfico de cada sitio peligroso.

```
CREATE TABLE HAZARDOUS_SITES ( site_id   integer,
                              name       varchar(128),
                               location  db2gse.ST_Point);
```

La consulta devuelve una lista de áreas sensibles y nombres de sitios peligrosos correspondientes a áreas sensibles que forman intersección con el área con un radio de cinco millas que rodea los sitios peligrosos.

```
SELECT sa.name, hs.name
FROM SENSITIVE_AREAS sa, HAZARDOUS_SITES hs
WHERE db2gse.ST_Intersects(db2gse.ST_Buffer(hs.location,(5 * 5280)),sa.zone)
= 1;
```

ST_IsClosed

ST_IsClosed toma una serie lineal o una geometría varias series lineales y devuelve 1 (TRUE) si está cerrada y 0 (FALSE) si no es así.

Sintaxis

db2gse.ST_IsClosed(c db2gse.ST_Curve)
db2gse.ST_IsClosed(mc db2gse.ST_MultiCurve)

Tipo devuelto

Entero

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla CLOSED_LINESTRING, que contiene una sola columna de series lineales.

```
CREATE TABLE CLOSED_LINESTRING (ln1 db2gse.ST_LineString)
```

Las siguientes sentencias INSERT insertan dos registros en la tabla CLOSED_LINESTRING. El primer registro no es una serie lineal cerrada y el segundo sí lo es.

```
INSERT INTO CLOSED_LINESTRING
VALUES(db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO CLOSED_LINESTRING
VALUES(db2gse.ST_LineFromText('linestring  (10.02 20.01,11.92 35.64,
       25.02 34.15, 19.15 33.94,10.02 20.01)',
                     db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados asociado muestran los resultados de la función ST_IsClosed. La primera fila devuelve un 0 porque la serie lineal no está cerrada, mientras que la segunda fila devuelve un 1 porque la serie lineal está cerrada.

```
SELECT db2gse.ST_IsClosed(ln1) "Is it closed" FROM CLOSED_LINESTRING

Is it closed
------------
           0
           1

  2 record(s) selected.
```

La siguiente sentencia CREATE TABLE crea la tabla CLOSED_MULTILINESTRING, que tiene una sola columna de geometrías varias series lineales.

```
CREATE TABLE CLOSED_MULTILINESTRING (mln1 db2gse.ST_MultiLineString)
```

Las siguientes sentencias INSERT insertan dos registros en CLOSED_MULTILINESTRING, un registro de geometría varias series lineales que no está cerrada y otra que sí lo está.

```
INSERT INTO CLOSED_MULTILINESTRING
VALUES(db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,10.32 23.98,
       11.92 25.64), (9.55 23.75,15.36 30.11))',
                      db2gse.coordref()..srid(0)))

INSERT INTO CLOSED_MULTILINESTRING
VALUES(db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,11.92 35.64,
       25.02 34.15, 19.15 33.94,10.02 20.01),
                                        (51.71 21.73,73.36 27.04,71.52 32.87,
                                         52.43 31.90,51.71 21.73))',
                      db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados asociado muestran los resultados de la función ST_IsClosed. La primera fila devuelve 0 porque la geometría varias series lineales no está cerrada, mientras que la segunda fila devuelve 1 porque la geometría varias series lineales está cerrada. Una geometría varias series lineales está cerrada si todos sus elementos series lineales están cerrados.

```
SELECT db2gse.ST_IsClosed(mln1) "Is it closed" FROM CLOSED_MULTILINESTRING

Is it closed
------------
           0
           1

  2 record(s) selected.
```

ST_IsEmpty

ST_IsEmpty toma un objeto geometría y devuelve 1 (TRUE) si está vacío y 0 (FALSE) si no es así.

Sintaxis

db2gse.ST_IsEmpty(g db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla EMPTY_TEST con dos columnas. La columna GEOTYPE almacena el tipo de datos de las subclases que se almacenan en la columna de geometría G1.

```
CREATE TABLE EMPTY_TEST (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Las siguientes sentencias INSERT insertan dos registros para las subclases de geometría punto, serie lineal y polígono. Un registro está vacío y el otro no.

```
INSERT INTO EMPTY_TEST
VALUES('Point', db2gse.ST_PointFromText('point (10.02 20.01)',
db2gse.coordref()..srid(0)))

INSERT INTO EMPTY_TEST
VALUES('Point', db2gse.ST_PointFromText('point empty',
db2gse.coordref()..srid(0)))

INSERT INTO EMPTY_TEST
VALUES('Linestring', db2gse.ST_LineFromText('linestring (10.02 20.01,
       10.32 23.98, 11.92 25.64)',
                                  db2gse.coordref()..srid(0)))

INSERT INTO EMPTY_TEST
VALUES('Linestring', db2gse.ST_LineFromText('linestring  empty',
db2gse.coordref()..srid(0)))

INSERT INTO EMPTY_TEST
VALUES('Polygon', db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,
25.02 34.15,19.15 33.94,10.02 20.01))',
                               db2gse.coordref()..srid(0)))

INSERT INTO EMPTY_TEST
VALUES('Polygon', db2gse.ST_PolyFromText('polygon  empty',
db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente muestran el tipo de geometría de la columna GEOTYPE y los resultados de la función ST_IsEmpty.

```
SELECT geotype, db2gse.ST_IsEmpty(g1) "It is empty" FROM EMPTY_TEST

GEOTYPE                   It is empty
--------------------      -----------
ST_Point                      0
ST_Point                    1
ST_Linestring                 0
ST_Linestring               1
ST_Polygon                    0
ST_Polygon                  1

6 record(s) selected.
```

ST_IsRing

ST_IsRing toma una serie lineal y devuelve 1 (TRUE) si es un anillo (es decir, la serie lineal está cerrada y es sencilla) y 0 (FALSE) si no es así.

Sintaxis

db2gse.ST_IsRing(c db2gse.ST_Curve)

Tipo devuelto

Entero

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla RING_LINESTRING, que tiene una sola columna de series lineales denominada LN1.

```
CREATE TABLE RING_LINESTRING (ln1 db2gse.ST_LineString)
```

Las siguientes sentencias INSERT insertan tres series lineales en la columna LN1. La primera fila contiene una serie lineal que no está cerrada y por lo tanto no es un anillo. La segunda fila contiene una serie lineal que está cerrada y es sencilla, por lo tanto es un anillo. La tercera fila contiene una serie lineal que está cerrada pero no es sencilla porque forma intersección con su propio interior, por lo tanto no es un anillo.

```
INSERT INTO RING_LINESTRING
VALUES(db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0)))

INSERT INTO RING_LINESTRING
VALUES(db2gse.ST_LineFromText('linestring (10.02 20.01,11.92 35.64,25.02 34.15,
                                19.15 33.94, 10.02 20.01)',
                    db2gse.coordref()..srid(0)))

INSERT INTO RING_LINESTRING
VALUES(db2gse.ST_LineFromText('linestring (15.47 30.12,20.73 22.12,10.83 14.13,
                                16.45 17.24,21.56 13.37,11.23 22.56,
                                19.11 26.78,15.47 30.12)',
                    db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente muestran los resultados de la función ST_IsRing. Las filas primera y tercera devuelven un 0. Esto se debe a que las series lineales no son anillos, mientras que la segunda fila devuelve un 1 porque es un anillo.

```
SELECT db2gse.ST_IsRing(ln1) "Is it ring" FROM RING_LINESTRING

Is it ring
-----------
          0
          1
          0

  3 record(s) selected.
```

ST_IsSimple

ST_IsSimple toma un objeto geometría y devuelve 1 (TRUE) si el objeto es sencillo y 0 (FALSE) si no lo es.

Sintaxis

db2gse.ST_IsSimple(g db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla ISSIMPLE_TEST, que tiene dos columnas. La columna PID, que es de tipo smallint, contiene el identificador exclusivo de cada fila. La columna de geometría G1 almacena los ejemplos de geometrías sencillas y no sencillas.

```
CREATE TABLE ISSIMPLE_TEST (pid smallint, g1 db2gse.ST_Geometry)
```

Las siguientes sentencias INSERT insertan dos registros en la tabla ISSIMPLE_TEST. El primero es sencillo porque es una serie lineal que no forma intersección con su interior. El segundo no es sencillo porque forma intersección con su interior.

```
INSERT INTO ISSIMPLE_TEST
VALUES (1, db2gse.ST_LineFromText('linestring (10 10, 20 20, 30 30)',
db2gse.coordref()..srid(0)))

INSERT INTO ISSIMPLE_TEST
VALUES (2, db2gse.ST_LineFromText('linestring (10 10, 20 20,20 30,10 30,10 20,
20 10)', db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente muestran los resultados de la función ST_IsSimple. El primer registro devuelve un 1 porque la serie lineal es sencilla, mientras que el segundo registro devuelve un 0 porque la serie lineal no es sencilla.

```
SELECT ST_IsSimple(g1)
FROM ISSIMPLE_TEST

g1
--------------
          1
          0
```

ST_IsValid

ST_IsValid toma un valor ST_Geometry y devuelve 1 (TRUE) si es válido y 0 (FALSE) si no lo es. Una geometría insertada en una base de datos DB2 siempre será válida porque DB2 Spatial Extender siempre valida sus datos espaciales antes de aceptarlos. Sin embargo, puede que otros proveedores de DBMS no validen la entrada, por lo que la aplicación debe hacerlo.

Sintaxis

db2gse.ST_IsValid(g db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

La tabla valid_test se crea con las columnas geotype y g1. La columna geotype almacena el nombre de la subclase de geometría almacenada en la columna de geometría g1.

```
CREATE TABLE valid_test (geotype varchar(20), g1 db2gse.ST_Geometry)
```

Las sentencias INSERT insertan una subclase de ejemplo en la tabla valid_test.

```
INSERT INTO valid_test VALUES(
    'Point', db2gse.ST_PointFromText('point (10.02 20.01)',
db2gse.coordref()..srid(0))
)

INSERT INTO valid_test VALUES(
'Linestring',
       db2gse.ST_LineFromText('linestring  (10.02 20.01,10.32 23.98,11.92 25.64)',
db2gse.coordref()..srid(0))
)

INSERT INTO valid_test VALUES(
'Polygon', db2gse.ST_PolyFromText('polygon ((10.02 20.01,11.92 35.64,
    25.02 34.15, 19.15 33.94,10.02 20.01))', db2gse.coordref()..srid(0))
)

INSERT INTO valid_test VALUES(
'Multipoint', db2gse.ST_MPointFromText('multipoint (10.02 20.01,10.32 23.98,
11.92 25.64)', db2gse.coordref()..srid(0))
)

INSERT INTO valid_test VALUES(
'Multilinestring', db2gse.ST_MLineFromText('multilinestring ((10.02 20.01,
10.32 23.98,11.92 25.64),(9.55 23.75,15.36 30.11))',
db2gse.coordref()..srid(0))
)

INSERT INTO valid_test VALUES(
'Multipolygon',
```

```
db2gse.ST_MPolyFromText('multipolygon (((10.02 20.01,11.92 35.64,
25.02 34.15,19.15 33.94,10.02 20.01)),((51.71 21.73,73.36 27.04,71.52 32.87,
52.43 31.90,51.71 21.73)))', db2gse.coordref()..srid(0))
)
```

La sentencia SELECT lista el nombre de subclase almacenado en la columna geotype con la dimensión de dicho tipo de geotype.

```
SELECT geotype, db2gse.ST_IsValid(g1) Valid FROM valid_test

GEOTYPE              Valid
-------------------- -------------
ST_Point                     1
ST_Linestring                1
ST_Polygon                   1
ST_Multipoint                1
ST_Multilinestring           1
ST_Multipolygon              1

6 record(s) selected.
```

ST_Length

ST_Length toma una geometría serie lineal o varias series lineales y devuelve su longitud.

Sintaxis

db2gse.ST_Length(c db2gse.ST_Curve)
db2gse.ST_Length(mc db2gse.ST_MultiCurve)

Tipo devuelto

Double

Ejemplos

Un ecologista local está estudiando los patrones de migración de la población de salmones de las vías fluviales de la región. El ecologista desea saber la longitud de todos los sistemas de ríos y corrientes que fluyen por la región.

La siguiente sentencia CREATE TABLE crea la tabla WATERWAYS. Las columnas ID y NAME identifican cada sistema de corrientes y de ríos que se almacena en la tabla. La columna WATER es una geometría varias series lineales porque los sistemas de ríos y de corrientes suelen ser un agregado de varias series lineales.

```
CREATE TABLE WATERWAYS (id integer, name varchar(128),
                            water      db2gse.ST_MultiLineString);
```

La siguiente sentencia SELECT utiliza la función ST_Length para devolver el nombre y la longitud de cada vía fluvial de la región.

```
SELECT name, db2gse.ST_Length(water) "Length"
FROM WATERWAYS;
```

La Figura 36 en la página 248 muestra los sistemas de ríos y corrientes que fluyen dentro de los límites de la región.

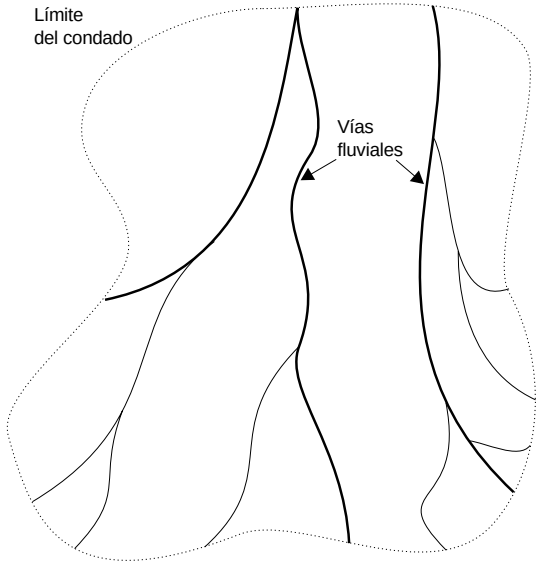


Figura 36. Utilización de ST_Length para determinar la longitud total de las vías fluviales de una región

ST_LineFromText

ST_LineFromText toma una representación de texto conocido de tipo serie lineal y una identidad del sistema de referencias espaciales y devuelve una serie lineal.

Sintaxis

db2gse.ST_LineFromText(lineStringTaggedText Varchar(4000), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_LineString

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla LINESTRING_TEST, que contiene una sola columna de series lineales LN1.

```
CREATE TABLE LINESTRING_TEST (ln1 db2gse.ST_LineString)
```

La siguiente sentencia INSERT inserta una serie lineal en la columna LN1 utilizando la función ST_LineFromText.

```
INSERT INTO LINESTRING_TEST
VALUES (db2gse.ST_LineFromText('linestring(10.01 20.03,20.94 21.34,
        35.93 19.04)', db2gse.coordref()..srid(0)))
```

ST_LineFromWKB

ST_LineFromWKB toma una representación de binario conocido del tipo serie lineal y una identidad del sistema de referencias espaciales y devuelve una serie lineal.

Sintaxis

db2gse.ST_LineFromWKB(WKBLineString Blob(1M), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_LineString

Ejemplos

El siguiente fragmento de código llena la tabla SEWERLINES con el id exclusivo, clase de tamaño y geometría de cada línea de alcantarillado.

La tabla SEWERLINES se crea con tres columnas. La primera columna, SEWER_ID, identifica de forma exclusiva cada línea de alcantarillado. La segunda columna, CLASS, de tipo entero, identifica el tipo de línea de alcantarillado, que suele estar asociado a la capacidad de la línea. La tercera columna, SEWER, de tipo serie lineal, almacena la geometría de la línea de alcantarillado.

```
CREATE TABLE SEWERLINES (sewer_id   integer,
                         class  integer,
                         sewer  db2gse.ST_LineString);

/* Crear la sentencia SQL insert para llenar el id de alcantarillado,
   la clase de tamaño y la serie lineal del alcantarillado.
   Los interrogantes son marcadores de parámetro que indican los valores
   de id de alcantarillado, clase y geometría de alcantarillado que se
   recuperarán en el momento de la ejecución. */
strcpy (wkb_sql,"insert into sewerlines (sewer_id,class,sewer)
values (?,?, db2gse.ST_LineFromWKB (cast(? as blob(1m)),
        db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Vincular el valor entero sewer_id con el primer parámetro. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_id, 0, &pcbvalue1);

/* Vincular el valor entero class con el segundo parámetro. */
pcbvalue2 = 0;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &sewer_class, 0, &pcbvalue2);
```

```
/* Vincular la forma con el tercer parámetro. */
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, sewer_wkb, blob_len, &pcbvalue3);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

ST_MLineFromText

ST_MLineFromText toma una representación de texto conocido de tipo varias series lineales y una identidad del sistema de referencias espaciales y devuelve una geometría varias series lineales.

Sintaxis

db2gse.ST_MLineFromText(multiLineStringTaggedText String, cr db2gse.coordref)

Tipo devuelto

db2gse.ST_MultiLineString

Ejemplos

la siguiente sentencia CREATE TABLE crea la tabla MLINESTRING_TEST. MLINESTRING_TEST tiene dos columnas: la columna tipo smallint GID, que identifica de forma exclusiva la fila, y la columna tipo varias series lineales ML1.

```
CREATE TABLE ST_MLINESTRING_TEST (gid smallint, ml1 db2gse.ST_MultiLineString)
```

La siguiente sentencia INSERT inserta la geometría varias series lineales con la función ST_MLineFromText.

```
INSERT INTO MLINESTRING_TEST
VALUES (1, db2gse.ST_MLineFromText('multilinestring((10.01 20.03,10.52 40.11,
         30.29 41.56,31.78 10.74),
                                        (20.93 20.81, 21.52 40.10))',
                         db2gse.coordref()..srid(0)))
```

ST_MLineFromWKB

ST_MLineFromWKB toma una representación de binario conocido de tipo varias series lineales y una identidad del sistema de referencias espaciales y devuelve una geometría varias series lineales.

Sintaxis

db2gse.ST_MLineFromWKB(WKBMultiLineString Blob(1M), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_MultiLineString

Ejemplos

El siguiente fragmento de código llena la tabla WATERWAYS con un id exclusivo, un nombre y una geometría varias series lineales denominada water.

La tabla WATERWAYS se crea con las columnas ID y NAME, que identifican cada sistema de corrientes y de ríos almacenado en la tabla. La columna WATER es una geometría varias series lineales porque los sistemas de ríos y de corrientes suelen ser un agregado de varias series lineales.

```
CREATE TABLE WATERWAYS (id     integer,
                        name       varchar(128),
                        water      db2gse.ST_MultiLineString);

/* Crear la sentencia SQL insert para llenar el id, nombre y
   varias series lineales. Los interrogantes son marcadores de parámetro que
   indican los valores de id, name y water que se recuperarán en el
   momento de la ejecución. */
strcpy (shp_sql,"insert into WATERWAYS (id,name,water)
values (?,?, db2gse.ST_MLineFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)shp_sql, SQL_NTS);

/* Vincular el valor entero id con el primer parámetro. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);

/* Vincular el valor varchar name con el segundo parámetro. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_CHAR, name_len, 0, &name, name_len, &pcbvalue2);

/* Vincular la forma con el tercer parámetro. */
```

```
pcbvalue3 = blob_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, water_shape, blob_len, &pcbvalue3);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

ST_MPointFromText

ST_MPointFromText toma una representación de texto conocido de tipo varios puntos y una identidad del sistema de referencias espaciales y devuelve una geometría varios puntos.

Sintaxis

db2gse.ST_MPointFromText(multiPointTaggedText Varchar(4000), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_MultiPoint

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla MULTIPOINT_TEST con una sola columna tipo varios puntos, MPT1.

```
CREATE TABLE MULTIPOINT_TEST (mpt1 db2gse.ST_MultiPoint)
```

La siguiente sentencia INSERT inserta una geometría varios puntos en la columna MPT1 utilizando la columna ST_MPointFromText.

```
INSERT INTO MULTIPOINT_TEST
VALUES (1, db2gse.ST_MPointFromText('multipoint(10.01 20.03,10.52 40.11,
                                    30.29 41.56,31.78 10.74)',
                         db2gse.coordref()..srid(0)))
```

ST_MPointFromWKB

ST_MPointFromWKB toma una representación de binario conocido de tipo varios puntos y una identidad del sistema de referencias espaciales y devuelve una geometría varios puntos.

Sintaxis

db2gse.ST_MPointFromWKB(WKBMultiPoint Blob(1M), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_MultiPoint

Ejemplos

El siguiente fragmento de código llena la tabla SPECIES_SITINGS.

La tabla SPECIES_SITINGS se crea con tres columnas. Las columnas SPECIES y GENUS identifican de forma exclusiva cada fila, mientras que la columna tipo varios puntos SITINGS almacena las ubicaciones de las localizaciones de especies.

```
CREATE TABLE SPECIES_SITINGS (species  varchar(32),
                              genus varchar(32),
                              sitings  db2gse.ST_MultiPoint);

/* Crear la sentencia SQL insert para llenar las columnas species, genus y
   sitings. Los interrogantes son marcadores de parámetro que indican los
   valores species, genus y sitings que se recuperarán en el momento de
   la ejecución. */
strcpy (wkb_sql,"insert into SPECIES_SITINGS (species,genus,sitings)
values (?,?, db2gse.ST_MPointFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Vincular el valor varchar species con el primer parámetro. */
pcbvalue1 = species_len;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, species_len, 0, &species, species_len, &pcbvalue1);
/* Vincular el valor varchar genus con el segundo parámetro. */
pcbvalue2 = genus_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, genus_len, 0, &name, genus_len, &pcbvalue2);

/* Vincular la forma con el tercer parámetro. */
pcbvalue3 = sitings_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, sitings_len, 0, sitings_wkb, sitings_len, &pcbvalue3);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

ST_MPolyFromText

ST_MPolyFromText toma una representación de texto conocido de tipo varios polígonos y una identidad del sistema de referencias espaciales y devuelve una geometría varios polígonos.

Sintaxis

db2gse.ST_MPolyFromText(multiPolygonTaggedText Varchar(4000), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_MultiPolygon

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla MULTIPOLYGON_TEST que tiene una sola columna tipo varios polígonos, MPL1.

```
CREATE TABLE MULTIPOLYGON_TEST (mpl1 db2gse.ST_MultiPolygon)
```

La siguiente sentencia INSERT inserta una geometría varios polígonos en la columna MPL1 utilizando la función ST_MPolyFromText.

```
INSERT INTO MULTIPOLYGON_TEST VALUES (
db2gse.ST_MPolyFromText('multipolygon(((10.01 20.03,10.52 40.11,
 30.29 41.56,31.78 10.74,10.01 20.03),(21.23 15.74,21.34 35.21,28.94 35.35,
 29.02 16.83, 21.23 15.74)),((40.91 10.92,40.56 20.19,
 50.01 21.12,51.34 9.81, 40.91 10.92)))',
 db2gse.coordref()..srid(0)))
```

ST_MPolyFromWKB

ST_MPolyFromWKB toma una representación de binario conocido de tipo varios polígonos y una identidad del sistema de referencias espaciales y devuelve una geometría varios polígonos.

Sintaxis

db2gse.ST_MPolyFromWKB(WKBMultiPolygon Blob(1M), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_MultiPolygon

Ejemplos

El siguiente fragmento de código llena la tabla LOTS.

La tabla LOTS almacena la columna LOT_ID, que identifica de forma exclusiva cada parcela, y la geometría varios polígonos LOT, que contiene la geometría de líneas de la parcela.

```
CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );

/* Crear la sentencia SQL insert para llenar las columnas lot_id y lot. Los
   interrogantes son marcadores de parámetro que indican los valores lot_id y lot
   que se recuperarán en el momento de la ejecución.*/
strcpy (wkb_sql,"insert into LOTS (lot_id,lot)
values (?, db2gse.ST_MPolyFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Vincular el valor entero lot_id con el primer parámetro. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Vincular la forma de parcela con el segundo parámetro. */
pcbvalue2 = lot_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, lot_len, 0, lot_wkb, lot_len, &pcbvalue2);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

ST_NumGeometries

ST_NumGeometries toma un grupo y devuelve el número de geometrías del grupo.

Sintaxis

db2gse.ST_NumGeometries(g db2gse.ST_GeomCollection)

Tipo devuelto

Entero

Ejemplos

El ingeniero municipal desea saber el número real de edificios asociados a cada área edificada.

Las áreas edificadas se almacenan en la tabla BUILDINGFOOTPRINTS que se creó con la siguiente sentencia CREATE TABLE.

```
CREATE TABLE BUILDINGFOOTPRINTS ( building_id integer,
                                        lot_id      integer,
                                  footprint   db2gse.ST_MultiPolygon);
```

La siguiente sentencia SELECT utiliza la función ST_NumGeometries para listar el BUILDING_ID que identifica de forma exclusiva cada edificio y el número de edificios de cada área edificada.

```
SELECT building_id, db2gse.ST_NumGeometries (footprint) "Number of buildings"
FROM BUILDINGFOOTPRINTS;
```

ST_NumInteriorRing

ST_NumInteriorRing toma un polígono y devuelve el número de anillos interiores que contiene.

Sintaxis

db2gse.NumInteriorRing(p db2gse.ST_Polygon)

Tipo devuelto

Entero

Ejemplos

Un ornitólogo que está estudiando la población de aves en varias islas de los mares del sur sabe que la zona en la que se alimenta una determinada especie se limita a las islas que contienen lagos de agua fresca. Por lo tanto, desea saber qué islas contienen uno o más lagos.

La siguiente sentencia CREATE TABLE crea la tabla ISLANDS. Las columnas ID y NAME de la tabla ISLANDS identifican cada isla y la columna tipo polígono LAND almacena la geometría de la isla.

```
CREATE TABLE ISLANDS (id integer, name varchar(32), land db2gse.ST_Polygon);
```

Puesto que los anillos interiores representan lagos, se utiliza la función ST_NumInteriorRing para listar únicamente las islas que tienen al menos un anillo interior.

```
SELECT name FROM ISLANDS WHERE db2gse.ST_NumInteriorRing(land) > 0;
```

ST_NumPoints

ST_NumPoints toma una serie lineal y devuelve el número de puntos que contiene.

Sintaxis

db2gse.ST_NumPoints(l db2gse.ST_LineString)

Tipo devuelto

Entero

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla NUMPOINTS_TEST. La columna GEOTYPE contiene el tipo de geometría almacenado en la columna de geometría G1.

```
CREATE TABLE NUMPOINTS_TEST (geotype varchar(12), g1 db2gse.ST_Geometry)
```

La siguiente sentencia INSERT inserta una serie lineal.

```
INSERT INTO NUMPOINTS_TEST VALUES( linestring,
db2gse.ST_LineFromText('linestring (10.02 20.01, 23.73 21.92)',
db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente listan el tipo de geometría y el número de puntos que contiene cada una.

```
SELECT geotype, db2gse.ST_NumPoints(g1)
FROM NUMPOINTS_TEST

GEOTYPE      Number of points
------------ ----------------
ST_linestring          2
  1 record(s) selected.
```

ST_OrderingEquals

ST_OrderingEquals compara dos geometrías y devuelve 1 (TRUE) si las geometrías son iguales y las coordenadas están en el mismo orden y 0 (FALSE) si no es así.

Sintaxis

db2gse.ST_OrderingEquals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla LINESTRING_TEST, que tiene dos columnas tipo serie lineal, L1 y L2.

```
CREATE TABLE LINESTRING_TEST (lid integer, l1 db2gse.ST_LineString,
l2 db2gse.ST_LineString);
```

La siguiente sentencia INSERT inserta en L1 y L2 dos series lineales que son iguales y tienen el mismo orden de coordenadas.

```
INSERT INTO linestring_test VALUES (1,
db2gse.LineFromText('linestring (10.01 20.02, 21.50 12.10)',
  db2gse.coordref()..srid(0)),
db2gse.LineFromText('linestring (10.01 20.02, 21.50 12.10)',
db2gse.coordref()..srid(0)));
```

La siguiente sentencia INSERT inserta en L1 y L2 dos series lineales que son iguales pero no tienen el mismo orden de coordenadas.

```
INSERT INTO linestring_test VALUES (2,
db2gse.LineFromText('linestring (10.01 20.02, 21.50 12.10)',
  db2gse.coordref()..srid(0)),
db2gse.LineFromText('linestring (21.50 12.10,10.01 20.02)',
db2gse.coordref()..srid(0)));
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente muestra cómo la función ST_Equals devuelve 1 (TRUE) independientemente del orden de las coordenadas. La función ST_OrderingEquals devuelve 0 (FALSE) si no se cumplen las dos condiciones siguientes: son iguales y tienen el mismo orden de coordenadas.

```
SELECT lid, db2gse.ST_Equals(l1,l2) equals, db2gse.ST_OrderingEquals(l1,l2)
OrderingEquals
FROM linestring_test

lid   equals   OrderingEquals
---   ------   -------------
 1       1            1
 2       1            0
```

ST_Overlaps

ST_Overlaps toma dos objetos de geometría y devuelve 1 (TRUE) si la intersección entre los objetos da como resultado un objeto de geometría de la misma dimensión pero que no es igual a ninguno de los dos objetos fuente; si no es así, devuelve 0 (FALSE).

Sintaxis

db2gse.ST_Overlaps(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

El supervisor de un condado necesita una lista de sitios de residuos peligrosos cuyo radio de cinco millas se solape con áreas sensibles.

La siguiente sentencia CREATE TABLE crea la tabla SENSITIVE_AREAS. La tabla SENSITIVE_AREAS contiene varias columnas que describen las instituciones amenazadas además de la columna ZONE, que almacena la geometría de polígono de la institución.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                               size     float,
                               type     varchar(10),
                               zone     db2gse.ST_Polygon);
```

La tabla HAZARDOUS_SITES almacena la identidad de los sitios en las columnas SITE_ID y NAME, mientras que la ubicación geográfica real de cada sitio se almacena en la columna de puntos LOCATION.

```
CREATE TABLE HAZARDOUS_SITES ( site_id   integer,
                              name      varchar(128),
                               location  db2gse.ST_Point);
```

En la siguiente sentencia SELECT, las tablas SENSITIVE_AREAS y HAZARDOUS_SITES se unen mediante la función ST_Overlaps. Devuelve 1 (TRUE) para todas las filas de la tabla SENSITIVE_AREAS cuyos polígonos de zona se solapan con el radio de cinco millas que rodea el punto de cada ubicación HAZARDOUS_SITES.

```
SELECT hs.name
FROM HAZARDOUS_SITES hs, SENSITIVE_AREAS sa
WHERE db2gse.ST_Overlaps (buffer(hs.location,(5 * 5280)),sa.zone) = 1;
```

En la Figura 37 en la página 265, el hospital y el colegio se solapan con el radio de cinco millas de dos zonas de residuos peligrosos del condado, mientras que la guardería no se solapa con ninguno.

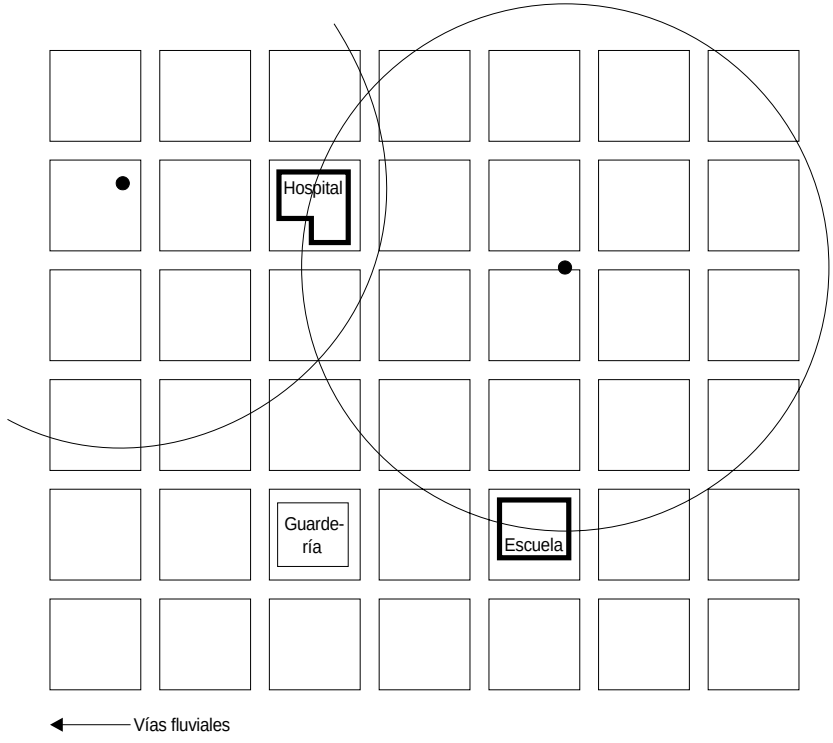


Figura 37. Utilización de ST_Overlaps para determinar los edificios que se encuentran, al menos parcialmente, dentro de un área de residuos peligrosos

ST_Perimeter

ST_Perimeter devuelve el perímetro de ST_Surface.

Sintaxis

db2gse.ST_Perimeter(s db2gse.ST_Surface)
db2gse.ST_Perimeter(ms db2gse.ST_MultiSurface)

Tipo devuelto

Double

Ejemplos

Un ecologista que estudia las aves que habitan en las orillas tiene que determinar la orilla de los lagos de una determinada área. Los lagos se almacenan como varios polígonos en la tabla WATERBODIES que se creó con la siguiente sentencia CREATE TABLE.

```
CREATE TABLE WATERBODIES (wbid  integer,
                          waterbody  db2gse.ST_MultiPolygon);
```

En la siguiente sentencia SELECT, la función ST_Perimeter devuelve el perímetro que rodea cada zona con agua, mientras que la función SUM suma los perímetros para devolver el total.

```
SELECT SUM(db2gse.ST_Perimeter(waterbody))
FROM waterbodies;
```

ST_PointFromText

ST_PointFromText toma una representación de texto conocido de tipo punto y una identidad del sistema de referencias espaciales y devuelve un punto.

Sintaxis

db2gse.ST_PointFromText(pointTaggedText Varchar(4000), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_Point

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla POINT_TEST, que contiene una sola columna de puntos, PT1.

```
CREATE TABLE POINT_TEST (pt1 db2gse.ST_Point)
```

Antes de que la sentencia INSERT inserte el punto en la columna PT1, la función ST_PointFromText convierte las coordenadas del texto del punto en el formato del punto.

```
INSERT INTO POINT_TEST VALUES (
       db2gse.ST_PointFromText ('point(10.01 20.03)',
         db2gse.coordref()..srid(0)))
```

ST_PointFromWKB

ST_PointFromWKB toma una representación de binario conocido de tipo punto y una identidad del sistema de referencias espaciales para devolver un punto.

Sintaxis

db2gse.ST_PointFromWKB(WKBPoint Blob(1M), srs SRID)

Tipo devuelto

db2gse.ST_Point

Ejemplos

El siguiente fragmento de código llena la tabla HAZARDOUS_SITES.

Los sitios peligrosos se almacenan en la tabla HAZARDOUS_SITES, creada con la siguiente sentencia CREATE TABLE. La columna LOCATION, definida como un punto, almacena la ubicación que constituye el centro geográfico de cada sitio peligroso.

```
CREATE TABLE HAZARDOUS_SITES ( site_id   integer,
                               name       varchar(128),
                                location  db2gse.ST_Point);

/* Crear la sentencia SQL insert para llenar las columnas site_id, name y
   location. Los interrogantes son marcadores de parámetro que indican los
   valores de site_id, name y location que se recuperarán en el momento de
   la ejecución. */
strcpy (wkb_sql,"insert into HAZARDOUS_SITES (site_id, name, location)
values (?,?, db2gse.ST_PointFromWKB(cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Vincular el valor entero site_id con el primer parámetro. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
     SQL_INTEGER, 0, 0, &site_id, 0, &pcbvalue1);

/* Vincular el valor varchar name con el segundo parámetro. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Vincular la forma location con el tercer parámetro. */
pcbvalue3 = location_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
```

```
     SQL_BLOB, location_len, 0, location_wkb, location_len, &pcbvalue3);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

ST_Point

ST_Point devuelve un ST_Point a partir de una coordenada y, una coordenada x y una referencia espacial determinadas.

Sintaxis

db2gse.ST_Point(X Double, Y Double, srs SRID)

Tipo devuelto

db2gse.ST_Point

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla POINT_TEST, que contiene una sola columna de puntos, PT1.

```
CREATE TABLE POINT_TEST (pt1 db2gse.ST_Point)
```

La función ST_Point convierte las coordenadas de punto en una geometría punto antes de que la sentencia INSERT lo inserte en la columna PT1.

```
INSERT INTO point_test VALUES(
    db2gse.ST_Point(10.01,20.03, db2gse.coordref()..srid(0))
)
```

ST_PointN

ST_PointN toma una serie lineal y un índice de entero y devuelve un punto que es el vértice número n de la ruta de la serie lineal.

Sintaxis

db2gse.ST_PointN(l db2gse.ST_Curve, n Integer)

Tipo devuelto

db2gse.ST_Point

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla POINTN_TEST, que tiene dos columnas: la columna GID, que identifica de forma exclusiva cada fila, y la columna tipo serie lineal LN1.

```
CREATE TABLE POINTN_TEST (gid integer, ln1 db2gse.ST_LineString)
```

Las siguientes sentencias INSERT insertan dos valores de serie lineal. La primera serie lineal no tiene coordenadas Z ni medidas, mientras que la segunda serie lineal tiene ambas.

```
INSERT INTO POINTN_TEST VALUES(1,
db2gse.ST_LineFromText('linestring (10.02 20.01,23.73 21.92,30.10 40.23)',
db2gse.coordref()..srid(0)))

INSERT INTO POINTN_TEST VALUES(2,
       db2gse.ST_LineFromText('linestring zm (10.02 20.01 5.0 7.0,23.73 21.92 6.5 7.1,
     30.10 40.23 6.9 7.2)', db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente listan la columna GID y el segundo vértice de cada serie lineal. La primera fila da como resultado un punto sin coordenada Z ni medida, mientras que la segunda fila da como resultado un punto con una coordenada Z y una medida. La función ST_PointN devuelve puntos con una coordenada Z o una medida si las hay en la serie lineal fuente.

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.ST_PointN(ln1,2)) AS varchar(60))
"The 2nd vertice"
FROM POINTN_TEST

GID         The 2nd vertice
----------- ------------------------------------------------------------
          1 POINT ( 23.73000000 21.92000000)
          2 POINT ZM ( 23.73000000 21.92000000 7.00000000 7.10000000)

  2 record(s) selected.
```

ST_PointOnSurface

ST_PointOnSurface toma un polígono o varios polígonos y devuelve an ST_Point.

Sintaxis

db2gse.ST_PointOnSurface(s db2gse.ST_Surface)
db2gse.ST_PointOnSurface(ms db2gse.ST_MultiSurface)

Tipo devuelto

db2gse.ST_Point

Ejemplos

El ingeniero municipal necesita crear un punto identificador para cada una de las áreas edificadas.

Las áreas edificadas se almacenan en la tabla BUILDINGFOOTPRINTS que se creó con la siguiente sentencia CREATE TABLE.

```
CREATE TABLE BUILDINGFOOTPRINTS ( building_id integer,
                                        lot_id      integer,
                                   footprint   db2gse.ST_MultiPolygon);
```

La función ST_PointOnSurface genera un punto que seguro que está en la superficie del área edificada. La función ST_PointOnSurface devuelve un punto que la función AsBinaryShape convierte en una forma forzada a una serie de caracteres de 1 megabyte para que la pueda utilizar la aplicación.

```
SELECT CAST(db2gse.AsBinaryShape(db2gse.ST_PointOnSurface(footprint)) as
            blob(1m))
FROM BUILDINGFOOTPRINTS;
```

ST_PolyFromText

ST_PolyFromText toma una representación de texto conocido de tipo polígono y una identidad del sistema de referencias espaciales y devuelve un polígono.

Sintaxis

db2gse.ST_PolyFromText(polygonTaggedText Varchar(4000), cr db2gse.coordref)

Tipo devuelto

db2gse.ST_Polygon

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla POLYGON_TEST con una sola columna de polígonos.

```
CREATE TABLE POLYGON_TEST (pl1 db2gse.ST_Polygon)
```

La siguiente sentencia INSERT inserta un polígono en la columna de polígonos mediante la función ST_PolyFromText.

```
INSERT INTO POLYGON_TEST VALUES (1,
db2gse.ST_PolyFromText('polygon((10.01 20.03,10.52 40.11,30.29 41.56,
    31.78 10.74,10.01 20.03))',
    db2gse.coordref()..srid(0)))
```

ST_PolyFromWKB

ST_PolyFromWKB toma una representación de binario conocido de tipo polígono y una identidad del sistema de referencias espaciales para devolver un polígono.

Sintaxis

db2gse.ST_PolyFromWKB(WKBPolygon Blob(1M), SRID Integer)

Tipo devuelto

db2gse.ST_Polygon

Ejemplos

El siguiente fragmento de código llena la tabla SENSITIVE_AREAS.

La tabla SENSITIVE_AREAS contiene varias columnas que describen las instituciones amenazadas además de la columna zone, que almacena la geometría de polígono de la institución.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                               size     float,
                               type     varchar(10),
                               zone     db2gse.ST_Polygon);


/* Crear la sentencia SQL insert para llenar las columnas
   id, name, size, type y zone. Los interrogantes son marcadores de parámetro
   que indican los valores id, name, size, type y zone que se recuperarán
   en el momento de la ejecución. */
strcpy (shp_wkb,"insert into SENSITIVE_AREAS (id, name, size, type, zone)
values (?,?,?,?, db2gse.ST_PolyFromWKB (cast(? as blob(1m)),
db2gse.coordref()..srid(0)))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */
rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */
rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Vincular el valor entero id con el primer parámetro. */
pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_INTEGER,
   SQL_INTEGER, 0, 0, &id, 0, &pcbvalue1);

/* Vincular el valor varchar name con el segundo parámetro. */
pcbvalue2 = name_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR,
   SQL_CHAR, 0, 0, name, 0, &pcbvalue2);

/* Vincular el valor float size con el tercer parámetro. */
pcbvalue3 = 0;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_FLOAT,
```

```
     SQL_REAL, 0, 0, &size, 0, &pcbvalue3);

/* Vincular el valor varchar type con el cuarto parámetro. */
pcbvalue4 = type_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_CHAR,
     SQL_VARCHAR, type_len, 0, type, type_len, &pcbvalue4);

/* Vincular el polígono zone con el quinto parámetro. */
pcbvalue5 = zone_len;
rc = SQLBindParameter (hstmt, 3, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, zone_len, 0, zone_wkb, zone_len, &pcbvalue5);

/* Ejecutar la sentencia insert. */
rc = SQLExecute (hstmt);
```

ST_Polygon

ST_Polygon genera un ST_Polygon a partir de un ST_LineString y un identificador del sistema de referencias espaciales.

Sintaxis

db2gse.ST_Polygon(l db2gse.ST_LineString, cr db2gse.coordref)

Tipo devuelto

db2gse.ST_Polygon

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla POLYGON_TEST, que tiene una sola columna, P1.

```
CREATE TABLE POLYGON_TEST (p1 db2gse.ST_polygon)
```

La siguiente sentencia INSERT convierte un anillo (una serie lineal cerrada y sencilla) en un polígono y lo inserta en la columna P1 mediante la función ST_LineFromText dentro de la función ST_Polygon.

```
INSERT INTO POLYGON_TEST VALUES (
db2gse.ST_Polygon(db2gse.ST_LineFromText('linestring(10.01 20.03,20.94
21.34,35.93 10.04,10.01 20.03)', db2gse.coordref()..srid(0))),
db2gse.coordref()..srid(0)))
)
```

ST_Relate

ST_Relate compara dos geometrías y devuelve 1 (TRUE) si las geometrías cumplen con las condiciones especificadas por la serie de la matriz patrón DE-9IM; si no es así, devuelve 0 (FALSE). Para obtener información sobre las matrices patrón DE-9IM, consulte el tema “Funciones de predicado” en la página 145.

Sintaxis

db2gse.ST_Relate(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry, patternMatrix String)

Tipo devuelto

Entero

Ejemplos

Una matriz patrón DE-9IM es un dispositivo que sirve para comparar geometrías. Hay varios tipos de matrices. Por ejemplo, la matriz patrón de *igualdad* le indicará si hay dos geometrías iguales.

En este ejemplo, una matriz patrón de igualdad, que se muestra en la Tabla 56, se lee de izquierda a derecha y de arriba a abajo como una serie (“T*F**FFF*”).

Tabla 56. Matriz patrón de igualdad

		b		
		Interior	**Límite**	**Exterior**
	Interior	T	*	F
a	**Límite**	*	*	F
	Exterior	F	F	*

Luego se crea la tabla RELATE_TEST con la siguiente sentencia CREATE TABLE.

```
CREATE TABLE RELATE_TEST (rid integer, g1 db2gse.ST_Geometry,
g2 db2gse.ST_Geometry, g3 db2gse.ST_Geometry);
```

Las siguientes sentencias INSERT insertan una subclase de ejemplo en la tabla RELATE_TEST.

```
INSERT INTO RELATE_TEST VALUES(
       1,
       db2gse.ST_PointFromText('point (10.02 20.01)',
            db2gse.coordref()..srid(0),
       db2gse.ST_PointFromText('point (10.02 20.01)',
            db2gse.coordref()..srid(0),
        db2gse.ST_PointFromText('point (30.01 20.01)',
            db2gse.coordref()..srid(0)
)
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente listan el nombre de subclase almacenado en la columna GEOTYPE con la dimensión de este tipo de geografía.

```
SELECT rid, relate(g1,g2) equals, relate(g1,g3) not_equals
FROM relate_test

RID     equals    not_equals
-------- -----------   -----------------
1         1              0

  1 record(s) selected.
```

ST_SRID

ST_SRID toma un objeto geometría y devuelve su identidad del sistema de referencias espaciales.

Sintaxis

db2gse.ST_SRID(g1 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

Durante la instalación de DB2 Spatial Extender se crea la tabla SPATIAL_REFERENCES. Cuando se crea una geometría, el SRID de dicha geometría se entra en la tabla SPATIAL_REFERENCES. La función ST_SRID devuelve el valor de dicha entrada.

Por ejemplo, se utiliza un tipo geometría en una sentencia CREATE TABLE:

```
CREATE TABLE SRID_TEST(g1 db2gse.ST_Geometry)
```

En la siguiente sentencia INSERT, se inserta una geometría de punto ubicada en la coordenada 10.01,50.76 en la columna de geometría G1. Cuando se creó la geometría de punto mediante la función ST_PointFromText, se le asignó el valor 1 de srid.

```
INSERT INTO SRID_TEST
VALUES (db2gse.ST_PointFromText('point(10.01 50.76)',
    db2gse.coordref()..srid(0)))
```

La función ST_SRID devuelve la identidad del sistema de referencias espaciales de la geometría que se acaba de entrar, tal como muestra la siguiente sentencia SELECT y el conjunto de resultados correspondiente.

```
SELECT db2gse.ST_SRID(g1) FROM SRID_TEST

g1
--------------
          1
```

ST_StartPoint

ST_StartPoint toma una serie lineal y devuelve un punto que es el primer punto de la serie lineal.

Sintaxis

db2gse.ST_StartPoint(c db2gse.ST_Curve)

Tipo devuelto

db2gse.ST_Point

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla STARTPOINT_TEST. STARTPOINT_TEST tiene dos columnas: la columna de enteros GID, que identifica de forma exclusiva las filas de la tabla, y la columna tipo serie lineal LN1.

```
CREATE TABLE STARTPOINT_TEST (gid integer, ln1 db2gse.ST_LineString)
```

Las siguientes sentencias INSERT insertan las series lineales en la columna LN1. La primera serie lineal no tiene coordenadas Z ni medidas, mientras que la segunda serie lineal tiene ambas.

```
INSERT INTO STARTPOINT_TEST VALUES(1,
db2gse.ST_LineFromText('linestring (10.02 20.01,23.73
21.92,30.10 40.23)', db2gse.coordref()..srid(0)))

INSERT INTO STARTPOINT_TEST VALUES(2,
db2gse.ST_LineFromText('linestring  zm (10.02 20.01 5.0 7.0,
    23.73 21.92 6.5 7.1,30.10 40.23 6.9 7.2)',
    db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente muestran cómo la función ST_StartPoint extrae el primer punto de cada serie lineal. La función ST_AsText convierte el punto en su formato de texto. El primer punto de la lista no tiene coordenada Z ni medida, mientras que el segundo punto tiene ambas porque la serie lineal fuente las tiene.

```
SELECT gid, CAST(db2gse.ST_AsText(db2gse.ST_StartPoint (ln1)) as varchar(60))
"Startpoint"
FROM STARTPOINT_TEST

GID         Startpoint
----------- ------------------------------------------------------------
          1 POINT ( 10.02000000 20.01000000)
          2 POINT ZM ( 10.02000000 20.01000000 5.00000000 7.00000000)

  2 record(s) selected.
```

ST_SymmetricDiff

ST_SymmetricDiff toma dos objetos geometría y devuelve un objeto geometría que es la diferencia simétrica de los objetos fuente.

Sintaxis

db2gse.ST_SymmetricDiff(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

El supervisor del condado debe determinar el área correspondiente a áreas sensibles que no forman intersección con el radio de cinco millas de sitios peligrosos.

La siguiente sentencia CREATE TABLE crea la tabla SENSITIVE_AREAS, que contiene varias columnas que describen las instituciones amenazadas. La tabla SENSITIVE_AREAS también contiene la columna ZONE, que almacena la geometría de polígono de cada institución.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                               size     float,
                               type     varchar(10),
                               zone     db2gse.ST_Polygon);
```

La siguiente sentencia CREATE TABLE crea la tabla HAZARDOUS_SITES, que almacena la identidad de los sitios en las columnas SITE_ID y NAME, mientras que la ubicación geográfica real de cada sitio se almacena en la columna de puntos LOCATION.

```
CREATE TABLE HAZARDOUS_SITES ( site_id   integer,
                              name       varchar(128),
                               location  point);
```

La función ST_Buffer genera un área de un radio de cinco millas que rodea las ubicaciones de los sitios de residuos peligrosos. La función ST_SymmetricDiff genera polígonos a partir de la intersección entre los polígonos de sitios de residuos peligrosos rodeados y las áreas sensibles. La función ST_Area devuelve el área de los polígonos de la intersección correspondiente a cada sitio peligroso.

```
SELECT sa.name, hs.name,
       db2gse.ST_Area(db2gse.ST_SymmetricDiff (db2gse.ST_Buffer(hs.location,
(5 * 5280)),sa.zone))
FROM HAZARDOUS_SITES hs, SENSITIVE_AREAS sa
```

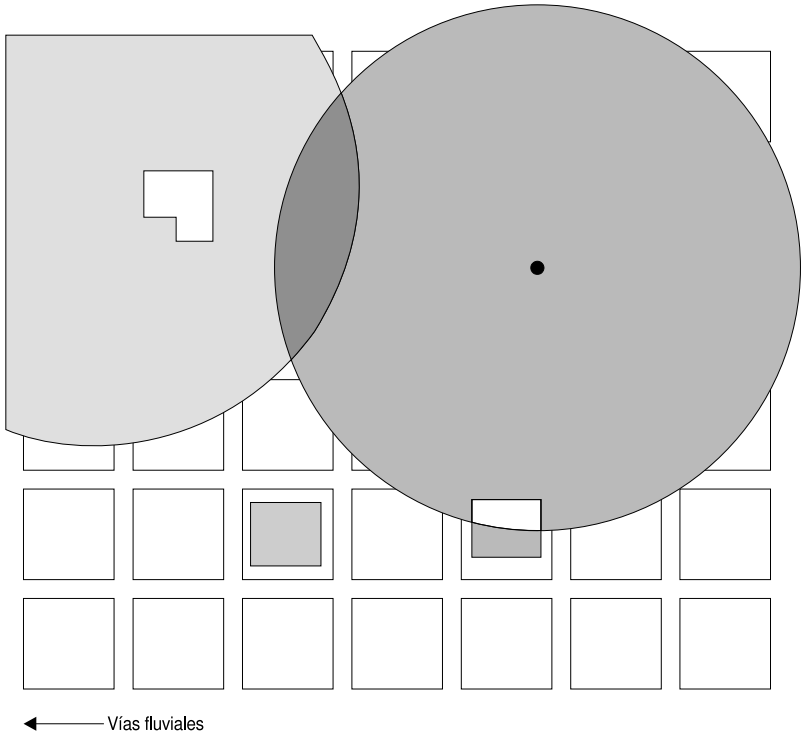


Figura 38. Utilización de ST_SymmetricDiff para determinar las áreas de residuos peligrosos que no contienen áreas sensibles (edificios habitados)

En la Figura 38, la diferencia simétrica de los sitios de residuos peligrosos y las áreas sensibles da como resultado la resta de las áreas que forman la intersección.

ST_Touches

ST_Touches devuelve 1 (TRUE) si ninguno de los puntos comunes a ambas geometrías forma intersección con los interiores de ambas geometrías; si no es así, devuelve 0 (FALSE). Al menos una geometría debe ser de tipo serie lineal, polígono, varias series lineales o varios polígonos.

Sintaxis

db2gse.ST_Touches(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

El técnico de GIS tiene que confeccionar una lista de líneas de alcantarillado cuyo final forma intersección con otra línea de alcantarillado.

La siguiente sentencia CREATE TABLE crea la tabla SEWERLINES, que tiene tres columnas. La primera columna, SEWER_ID, identifica de forma exclusiva cada línea de alcantarillado. La segunda columna, CLASS, de tipo entero, identifica el tipo de línea de alcantarillado, que suele estar asociado a la capacidad de la línea. La tercera columna, SEWER, de tipo serie lineal, almacena la geometría de la línea de alcantarillado.

```
CREATE TABLE SEWERLINES (sewer_id integer, class integer, sewer
    db2gse.ST_LineString);
```

La siguiente sentencia SELECT devuelve una lista ordenada de SEWER_ID que se tocan entre sí.

```
SELECT s1.sewer_id, s2.sewer_id
FROM sewerlines s1, sewerlines s2
WHERE db2gse.ST_Touches (s1.sewer, s2.sewer) = 1,
ORDER BY 1,2;
```

ST_Transform

ST_Transform asigna una geometría a un sistema de referencias espaciales que no es el sistema de referencias espaciales al que está asignada actualmente la geometría.

Sintaxis

db2gse.ST_Transform(g db2gse.ST_Geometry, cr db2gse.coordref)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla TRANSFORM_TEST, que tiene dos columnas tipo serie lineal, L1 y L2.

```
CREATE TABLE TRANSFORM_TEST (tid integer, l1 db2gse.ST_LineString,
l2 db2gse.ST_LineString)
```

La siguiente sentencia INSERT inserta una serie lineal en l1 con un SRID igual a 102.

```
INSERT INTO TRANSFORM_TEST VALUES (1, db2gse.ST_LineFromText('linestring
    (10.01 40.43, 92.32 29.89)',
    db2gse.coordref()..srid(102)),NULL)
```

La función ST_Transform convierte la serie lineal de L1 de la referencia de coordenadas asignada a SRID 102 a la referencia de coordenadas asignada a SRID 105. La siguiente sentencia UPDATE almacena la serie lineal transformada en la columna l2.

```
UPDATE TRANSFORM_TEST SET l2 = db2gse.ST_Transform(l1,
    db2gse.coordref()..srid(105))
```

ST_Union

ST_Union toma dos objetos geometría y devuelve un objeto geometría que es la unión de los objetos fuente.

Sintaxis

db2gse.ST_Union(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla SENSITIVE_AREAS, que contiene varias columnas que describen las instituciones amenazadas. La tabla SENSITIVE_AREAS también contiene la columna ZONE, que almacena la geometría de polígono de cada institución.

```
CREATE TABLE SENSITIVE_AREAS (id        integer,
                              name      varchar(128),
                               size     float,
                               type     varchar(10),
                               zone     db2gse.ST_Polygon);
```

La siguiente sentencia CREATE TABLE crea la tabla HAZARDOUS_SITES, que almacena la identidad de los sitios en las columnas SITE_ID y NAME. La ubicación geográfica real de cada sitio se almacena en la columna de puntos LOCATION.

```
CREATE TABLE HAZARDOUS_SITES (site_id integer, name varchar(128),
location db2gse.ST_Point);
```

La siguiente sentencia SELECT utiliza la función ST_Buffer para rodear con un radio de cinco millas las ubicaciones de sitios de residuos peligrosos. La función ST_Union genera polígonos a partir de la unión de los polígonos rodeados de sitios de residuos peligrosos y las áreas sensibles. La función ST_Area devuelve la unión de las áreas de los polígonos.

```
SELECT sa.name, hs.name,
       db2gse.ST_Area(db2gse.ST_Union(db2gse.ST_Buffer(hs.location,
(5 * 5280)),sa.zone))
FROM HAZARDOUS_SITES hs, SENSITIVE_AREAS sa;
```

ST_Within

ST_Within toma dos objetos geometría y devuelve 1 (TRUE) si el primer objeto queda completamente dentro del segundo y 0 (FALSE) si no es así.

Sintaxis

db2gse.ST_Within(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Tipo devuelto

Entero

Ejemplos

En el ejemplo siguiente, se crean dos tablas. La primera tabla, BUILDINGFOOTPRINTS, contiene las zonas edificadas de una ciudad. La segunda tabla, LOTS, contiene las parcelas de la ciudad. El ingeniero municipal desea asegurarse de que todas las áreas edificadas quedan completamente dentro de sus parcelas.

En ambas tablas, el tipo de datos varios polígonos almacena la geometría de las áreas edificadas y sus parcelas. El diseñador de la base de datos ha seleccionado el tipo varios polígonos para ambas funciones puesto que las parcelas pueden estar separadas por funciones naturales, como un río, y las áreas edificadas pueden costar de varios edificios.

```
CREATE TABLE BUILDINGFOOTPRINTS ( building_id integer,
                                        lot_id      integer,
                                   footprint   db2gse.ST_MultiPolygon);


CREATE TABLE LOTS (  lot_id integer, lot db2gse.ST_MultiPolygon );
```

Mediante la siguiente sentencia SELECT, el ingeniero municipal selecciona primero los edificios que no quedan completamente dentro de una parcela.

```
SELECT building_id
  FROM BUILDINGFOOTPRINTS, LOTS
 WHERE db2gse.ST_Within(footprint,lot) = 0;
```

Aunque la primera consulta ofrecerá una lista de todos los BUILDING_ID que tienen áreas edificadas fuera de un polígono de parcela, no determinará si el resto tienen asignado el lot_id correcto. Esta segunda sentencia SELECT realiza una comprobación de integridad de los datos de la columna LOT_ID de la tabla BUILDINGFOOTPRINTS.

```
SELECT bf.building_id "building id",
       bf.lot_id "buildings lot_id",
       LOTS.lot_id "LOTS lot_id"
  FROM BUILDINGFOOTPRINTS bf, LOTS
 WHERE db2gse.ST_Within(footprint,lot) = 1 AND
       LOTS.lot_id <> bf.lot_id;
```

ST_WKBToSQL

ST_WKBToSQL genera un valor ST_Geometry a partir de su representación de binario conocido. El valor SRID de 0 se utiliza automáticamente.

Sintaxis

db2gse.ST_WKBToSQL(WKBGeometry Blob(1M))

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla LOTS, que tiene dos columnas: la columna LOT_ID, que identifica de forma exclusiva cada parcela, y la columna de varios polígonos LOT, que contiene la geometría de cada parcela.

```
CREATE TABLE lots (lot_id  integer,
                   lot     db2gse.ST_MultiPolygon);
```

El siguiente fragmento de código C contiene funciones ODBC intercaladas con funciones SQL de DB2 Spatial Extender que insertan datos en la tabla LOTS.

La función ST_WKBToSQL convierte representaciones WKB en geometría de DB2 Spatial Extender. La sentencia INSERT completa se copia en una serie de caracteres wkb_sql. La sentencia INSERT contiene marcadores de parámetro para aceptar los datos LOT_ID y los datos LOT de forma dinámica.

```
/* Crear la sentencia SQL insert para llenar el id de parcela y
  los varios polígonos de parcela. Los interrogantes son marcadores de parámetro
  que indican los valores de lot_id y lot que se recuperarán en el momento
  de la ejecución. */

strcpy (wkb_sql,"insert into lots (lot_id, lot)
  values(?, db2gse.ST_WKBToSQL(cast(? as blob(1m))))");

/* Asignar memoria para el manejador de sentencias SQL y asociar
   el manejador de sentencias con el manejador de conexiones. */

rc = SQLAllocStmt (handle, &hstmt);

/* Preparar la sentencia SQL para la ejecución. */

rc = SQLPrepare (hstmt, (unsigned char *)wkb_sql, SQL_NTS);

/* Vincular el valor de clave de entero con el primer parámetro. */

pcbvalue1 = 0;
rc = SQLBindParameter (hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
     SQL_INTEGER, 0, 0, &lot_id, 0, &pcbvalue1);

/* Vincular la forma con el segundo parámetro. */
```

```
pcbvalue2 = blob_len;
rc = SQLBindParameter (hstmt, 2, SQL_PARAM_INPUT, SQL_C_BINARY,
     SQL_BLOB, blob_len, 0, shape_blob, blob_len, &pcbvalue2);

/* Ejecutar la sentencia insert. */

rc = SQLExecute (hstmt);
```

ST_WKTToSQL

ST_WKTToSQL genera un valor ST_Geometry a partir de su representación de texto conocido. El valor SRID de 0 se utiliza automáticamente.

Sintaxis

db2gse.ST_WKTToSQL(geometryTaggedText Varchar(4000))

Tipo devuelto

db2gse.ST_Geometry

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla GEOMETRY_TEST que contiene dos columnas: la columna GID de tipo entero, que identifica de forma exclusiva cada fila, y la columna G1, que almacena la geometría.

```
CREATE TABLE GEOMETRY_TEST (gid smallint, g1 db2gse.ST_Geometry)
```

Las siguientes sentencias INSERT insertan los datos en las columnas GID y G1 de la tabla GEOMETRY_TEST. La función ST_WKTToSQL convierte la representación de texto de cada geometría en su correspondiente subclase de DB2 Spatial Extender de la que se pueden crear instancias.

```
INSERT INTO GEOMETRY_TEST VALUES(
1, db2gse.ST_WKTToSQL ('point (10.02 20.01)')
)

INSERT INTO GEOMETRY_TEST VALUES(
2, db2gse.ST_WKTToSQL('linestring (10.01 20.01, 10.01 30.01, 10.01 40.01)')
)

INSERT INTO GEOMETRY_TEST VALUES(
3, db2gse.ST_WKTToSQL('polygon ((10.02 20.01, 11.92 35.64, 25.02 34.15,
19.15 33.94, 10.02 20.01))')
)

INSERT INTO GEOMETRY_TEST VALUES(
4, db2gse.ST_WKTToSQL('multipoint (10.02 20.01,10.32 23.98,11.92 25.64)')
)

INSERT INTO GEOMETRY_TEST VALUES(
5, db2gse.ST_WKTToSQL('multilinestring ((10.02 20.01,      10.32 23.98,
11.92 25.64),(9.55 23.75,15.36 30.11))')
)

INSERT INTO GEOMETRY_TEST VALUES(
6, db2gse.ST_WKTToSQL('multipolygon (((10.02 20.01, 11.92 35.64,
25.02 34.15, 19.15 33.94,10.02 20.01)),
((51.71 21.73, 73.36 27.04, 71.52 32.87, 52.43 31.90, 51.71 21.73)))')
)
```

ST_X

ST_X toma un punto y devuelve su coordenada X.

Sintaxis

ST_X(p ST_Point)

Tipo devuelto

Double

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla X_TEST, que tiene dos columnas: la columna GID, que identifica de forma exclusiva la fila, y la columna de puntos PT1.

```
CREATE TABLE X_TEST (gid integer, pt1 db2gse.ST_Point)
```

Las siguientes sentencias INSERT insertan dos filas. Una es un punto sin coordenada Z ni medida. La otra columna tiene una coordenada Z y una medida.

```
INSERT INTO X_TEST VALUES(1,
db2gse.ST_PointFromText('point (10.02 20.01)', db2gse.coordref()..srid(0)))

INSERT INTO X_TEST VALUES(2,
db2gse.ST_PointFromText('point  zm (10.02 20.01 5.0 7.0)',
    db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente listan la columna GID y la coordenada X doble de los puntos.

```
SELECT gid, db2gse.ST_X(pt1) "The X coordinate" FROM X_TEST

GID         The X coordinate
----------- ------------------------
          1   +1.00200000000000E+001
          2   +1.00200000000000E+001

  2 record(s) selected.
```

ST_Y

ST_Y toma un punto y devuelve su coordenada Y.

Sintaxis

db2gse.ST_Y(p db2gse.ST_Point)

Tipo devuelto

Double

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla Y_TEST, que tiene dos columnas: la columna GID, que identifica de forma exclusiva la fila, y la columna de puntos PT1.

```
CREATE TABLE Y_TEST (gid integer, pt1 db2gse.ST_Point)
```

Las sentencias INSERT insertan dos filas. Una es un punto sin coordenada Z ni medida. La otra columna tiene una coordenada Z y una medida.

```
INSERT INTO Y_TEST VALUES(1,
db2gse.ST_PointFromText('point (10.02 20.01)', db2gse.coordref()..srid(0)))

INSERT INTO Y_TEST VALUES(2,
db2gse.ST_PointFromText('point  zm (10.02 20.01 5.0 7.0)',
    db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente listan la columna GID y la coordenada Y doble de los puntos.

```
SELECT gid, db2gse.ST_Y(pt1) "The Y coordinate" FROM Y_TEST

GID         The Y coordinate
----------- ------------------------
          1   +2.00100000000000E+001
          2   +2.00100000000000E+001

  2 record(s) selected.
```

Z

Z toma un punto y devuelve su coordenada Z.

Sintaxis

Z(p db2gse.ST_Point)

Tipo devuelto

Double

Ejemplos

La siguiente sentencia CREATE TABLE crea la tabla Z_TEST con dos columnas: la columna GID, que identifica de forma exclusiva la fila, y la columna de puntos PT1.

```
CREATE TABLE Z_TEST (gid integer, pt1 db2gse.ST_Point)
```

Las siguientes sentencias INSERT insertan dos filas. Una es un punto sin coordenada Z ni medida. La otra columna tiene una coordenada Z y una medida.

```
INSERT INTO Z_TEST VALUES(1,
db2gse.ST_PointFromText('point (10.02 20.01)', db2gse.coordref()..srid(0)))

INSERT INTO Z_TEST VALUES(2,
db2gse.ST_PointFromText('point  zm (10.02 20.01 5.0 7.0)',
    db2gse.coordref()..srid(0)))
```

La siguiente sentencia SELECT y el conjunto de resultados correspondiente listan la columna GID y la coordenada Z doble de los puntos. La primera fila es NULA porque el punto no tiene coordenada Z.

```
SELECT gid, db2gse.Z(pt1) "The Z coordinate" FROM Z_TEST

GID         The Z coordinate
----------- ------------------------
          1                         -
          2   +5.00000000000000E+000

  2 record(s) selected.
```

Capítulo 15. Sistemas de coordenadas

Este capítulo contiene información de consulta sobre el sistema de referencias espaciales (SRS) y los valores de coordenadas que se utilizan para interpretar datos espaciales.

Visión general de los sistemas de coordenadas

La representación de texto conocido de sistemas de referencias espaciales ofrece una representación textual estándar para la información de los sistemas de referencias espaciales. Las definiciones de la representación de texto conocido se basan en el modelo de datos del sistema de coordenadas POSC/EPSG.

Un sistema de referencias espaciales es un sistema de coordenadas geográficas (latitud-longitud), proyectadas (X,Y) o geocéntricas (X,Y,Z). El sistema de coordenadas consta de varios objetos. Cada objeto tiene una palabra clave en mayúsculas (por ejemplo, DATUM o UNIT) seguida de los parámetros separados por comas que definen el objeto entre paréntesis. Algunos objetos están compuestos por otros objetos, así que el resultado es una estructura anidada.

Nota: Las implantaciones pueden sustituir los paréntesis estándares ( ) por corchetes [ ] y deben estar preparadas para leer ambos formatos de paréntesis.

La definición de EBNF (Extended Backus Naur Form) correspondiente a la representación de series de un sistema de coordenadas que utiliza corchetes es la siguiente (consulte la nota anterior sobre el uso de paréntesis):

```
<sistema coordenadas> = <sc proyectadas> | <sc geográficas> | <sc geocéntricas>
<sc proyectadas> = PROJCS["<nombre>", <sc geográficas>, <proyección>, {<parámetro>,}*
                <unidad lineal>]
<proyección> = PROJECTION["<nombre>"]
<parámetro> = PARAMETER["<nombre>", <valor>]
<valor> = <número>
```

Un sistema de coordenadas de un conjunto de datos se identifica mediante la palabra clave PROJCS si los datos están en coordenadas proyectadas (mediante GEOGCS si están en coordenadas geográficas o mediante GEOCCS si están en coordenadas geocéntricas). La palabra clave PROJCS va seguida de todas las ″piezas″ que definen el sistema de coordenadas proyectadas. La primera pieza de cualquier objeto es siempre el nombre. Tras el nombre del sistema de coordenadas proyectadas hay varios objetos: el sistema de coordenadas geográficas, la proyección de correlación, uno o más parámetros y la unidad lineal de medida. Todos los sistemas de coordenadas proyectadas se basan en un sistema de coordenadas geográficas, de modo que esta sección describe primero las piezas específicas de un sistema de coordenadas proyectadas. Por ejemplo, la zona UTM 10N del dato NAD83 se define del siguiente modo:

```
PROJCS["NAD_1983_UTM_Zone_10N",
<geographic cs>,
PROJECTION["Transverse_Mercator"],
PARAMETER["False_Easting",500000.0],
PARAMETER["False_Northing",0.0],
PARAMETER["Central_Meridian",-123.0],
PARAMETER["Scale_Factor",0.9996],
PARAMETER["Latitude_of_Origin",0.0],
UNIT["Meter",1.0]]
```

El nombre y varios objetos definen el objeto sistema de coordenadas geográficas: el dato, el meridiano principal y la unidad angular de medida.

```
<sc geográficas> = GEOGCS["<nombre>", <dato>, <meridiano principal>, <unidad angular>]
<dato> = DATUM["<nombre>", <esferoide>]
<esferoide> = SPHEROID["<nombre>", <eje semi-principal>, <allanamiento inverso>]
<eje semi-principal> = <número>
    (eje semi-principal se mide en metros y debe ser > 0.)
<allanamiento inverso> = <número>
<meridiano principal> = PRIMEM["<nombre>", <longitud>]
<longitud> = <número>
```

La serie del sistema de coordenadas geográficas para la zona UTM 10 en NAD83:

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],
UNIT["Degree",0.0174532925199433]]
```

El objeto UNIT puede representar unidades de medida angulares o lineales:

```
<unidad angular> = <unidad>
<unidad lineal> = <unidad>
<unidad> = UNIT["<nombre>", <factor conversión>]
<factor conversión> = <número>
```

El factor de conversión especifica número de metros (para una unidad lineal) o número de radianes (para una unidad angular) por unidad y debe ser mayor que cero.

De este modo, la representación completa de la serie de Zona UTM 10N es la siguiente:

```
PROJCS["NAD_1983_UTM_Zone_10N",
GEOGCS["GCS_North_American_1983",
DATUM[ "D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433]],
PROJECTION["Transverse_Mercator"],PARAMETER["False_Easting",500000.0],
PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",-123.0],
PARAMETER["Scale_Factor",0.9996],PARAMETER["Latitude_of_Origin",0.0],
UNIT["Meter",1.0]]
```

Un sistema de coordenadas geocéntricas es bastante parecido a un sistema de coordenadas geográficas:

```
<sc geocéntricas> = GEOCCS["<nombre>", <dato>, <meridiano principal>, <unidad lineal>]
```

Unidades lineales soportadas

Tabla 57. Unidades lineales soportadas

Unidad	Factor de conversión
Metro	1,0
Pie (Internacional)	0,3048
Pie EE.UU.	12 / 39,37
Pie americano modificado	12,0004584 / 39,37
Pie de Clarke	12 / 39,370432
Pie Indian	12 / 39,370141
Link	7,92 / 39,370432
Link (Benoit)	7,92 / 39,370113
Link (Sears)	7,92 / 39,370147
Chain (Benoit)	792 / 39,370113
Chain (Sears)	792 / 39,370147
Yarda (Indian)	36 / 39,370141
Yarda (Sears)	36 / 39,370147

Tabla 57. Unidades lineales soportadas (continuación)

Unidad	Factor de conversión
Braza	1,8288
Milla náutica	1852,0

Unidades angulares soportadas

Tabla 58. Unidades angulares soportadas

Unidad	Factor de conversión
Radián	1,0
Grado decimal	p / 180
Minuto decimal	(p / 180) / 60
Segundo decimal	(p / 180) / 36000
Gon	p / 200
Grado centesimal	p / 200

Esferoides soportados

Tabla 59. Esferoides soportados

Nombre	Eje semi-principal	Allanamiento inverso
Airy	6377563,396	299,3249646
Modified Airy	6377340,189	299,3249646
Australian	6378160	298,25
Bessel	6377397,155	299,1528128
Modified Bessel	6377492,018	299,1528128
Bessel (Namibia)	6377483,865	299,1528128
Clarke 1866	6378206,4	294,9786982
Clarke 1866 (Michigan)	6378693,704	294,978684677
Clarke 1880	6378249,145	293,465
Clarke 1880 (Arc)	6378249,145	293,466307656
Clarke 1880 (Benoit)	6378300,79	293,466234571
Clarke 1880 (IGN)	6378249,2	293,46602
Clarke 1880 (RGS)	6378249,145	293,465
Clarke 1880 (SGA)	6378249,2	293,46598

Tabla 59. Esferoides soportados (continuación)

Nombre	Eje semi-principal	Allanamiento inverso
Everest 1830	6377276,345	300,8017
Everest 1975	6377301,243	300,8017
Everest (Sarawak y Sabah)	6377298,556	300,8017
Modified Everest 1948	6377304,063	300,8017
Fischer 1960	6378166	298,3
Fischer 1968	6378150	298,3
Modified Fischer (1960)	6378155	298,3
GEM10C	6378137	298,257222101
GRS 1980	6378137	298,257222101
Hayford 1909	6378388	297,0
Helmert 1906	6378200	298,3
Hough	6378270	297,0
Internacional 1909	6378388	297,0
Internacional 1924	6378388	297,0
Nuevo internacional 1967	6378157,5	298,2496
Krasovsky	6378245	298,3
Mercury 1960	6378166	298,3
Modified Mercury 1968	6378150	298,3
NWL9D	6378145	298,25
OSU_86F	6378136,2	298,25722
OSU_91A	6378136,3	298,25722
Plessis 1817	6376523	308,64
South American 1969	6378160	298,25
Southeast Asia	6378155	298,3
Sphere (radius = 1.0)	1	0
Sphere (radius = 6371000 m)	6371000	0
Sphere (radius = 6370997 m)	6370997	0
Struve 1860	6378297	294,73
Walbeck	6376896	302,78
War Office	6378300,583	296
WGS 1960	6378165	298,3

Tabla 59. Esferoides soportados (continuación)

Nombre	Eje semi-principal	Allanamiento inverso
WGS 1966	6378145	298,25
WGS 1972	6378135	298,26
WGS 1984	6378137	298,257223563

Datos geodésicos soportados

Tabla 60. Datos geodésicos soportados

Adindan	Lisbon
Afgooye	Loma Quintana
Agadez	Lome
Australian Geodetic Datum 1966	Luzon 1911
Australian Geodetic Datum 1984	Mahe 1971
Ain el Abd 1970	Makassar
Amersfoort	Malongo 1987
Aratu	Manoca
Arc 1950	Massawa
Arc 1960	Merchich
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Bogota	Nord de Guerre
Bukit Rimpah	NGO 1948
Camacupa	Nord Sahara 1959
Campo Inchauspe	NSWC 9Z-2
Cape	Nouvelle Triangulation Francaise

Tabla 60. Datos geodésicos soportados (continuación)

Carthage	New Zealand Geodetic Datum 1949
Chua	OS (SN) 1980
Conakry 1905	OSGB 1936
Corrego Alegre	OSGB 1970 (SN)
Cote d'Ivoire	Padang 1884
Datum 73	Palestine 1923
Deir ez Zor	Pointe Noire
Deutsche Hauptdreiecksnetz	Provisional South American Datum 1956
Douala	Pulkovo 1942
European Datum 1950	Qatar
European Datum 1987	Qatar 1948
Egypt 1907	Qornoq
European Reference System 1989	RT38
Fahud	South American Datum 1969
Gandajika 1970	Sapper Hill 1943
Garoua	Schwarzeck
Geocentric Datum of Australia 1994	Segora
Guyane Francaise	Serindung
Herat North	Stockholm 1938
Hito XVIII 1963	Sudan
Hu Tzu	Shan Tananarive 1925
Hungarian Datum 1972	Timbalai 1948
Indian 1954	TM65
Indian 1975	TM75
Indonesian Datum 1974	Tokyo
Jamaica 1875	Trinidad 1903
Jamaica 1969	Trucial Coast 1948
Kalianpur	Voirol 1875
Kandawala	Voirol Unifie 1960
Kertau	WGS 1972
Kuwait Oil Company	WGS 1972 Transit Broadcast Ephemeris
La Canoa	WGS 1984
Lake	Yacare

Tabla 60. Datos geodésicos soportados (continuación)

Leigon	Yoff
Liberia 1964	Zanderij

Meridianos principales soportados

Tabla 61. Meridianos principales soportados

Ubicación	Coordenadas
Greenwich	0° 0' 0"
Berna	7° 26' 22,5" E
Bogotá	74° 4' 51,3" W
Bruselas	4° 22' 4,71" E
Ferro	17° 40' 0" W
Yakarta	106° 48' 27,79" E
Lisboa	9° 7' 54,862" W
Madrid	3° 41' 16,58" W
París	2° 20' 14,025" E
Roma	12° 27' 8,4" E
Estocolmo	18° 3' 29" E

Proyecciones de correlaciones soportadas

Tabla 62. Proyecciones de correlaciones soportadas

Proyecciones cilíndricas	Proyecciones seudocilíndricas
Behrmann	Parabólica de Craster
Cassini	Eckert I
Área igual cilíndrica	Eckert II
Equirectangular	Eckert III
Estereográfica de Gall	Eckert IV
Gauss-Kruger	Eckert V
Mercator	Eckert VI
Cilíndrica de Miller	Cuártico polar plano de McBryde-Thomas
Oblicua	Mercator (Hotine) Mollweide
Plate-Carée	Robinson

Tabla 62. Proyecciones de correlaciones soportadas (continuación)

Proyecciones cilíndricas	Proyecciones seudocilíndricas
Times	Sinusoidal (Sansom-Flamsteed)
Transverse Mercator	Winkel I

Proyecciones cónicas

Tabla 63. Proyecciones cónicas

Equivalente cónica de Albers	Trimétrica de Chamberlin
Cónica conforme oblicua bipolar	Equidistante a dos puntos
Bonne	Equivalente de Hammer-Aitoff
Cónica equidistante	Van der Grinten I
Cónica conforme Lambert	Variada
Policónica	Alaska serie E
Cónica simple	Alaska Grid (Estereográfica modificada por Snyder)

Proyecciones azimutales o planares

- Equidistante azimutal
- Perspectiva general lado interior vertical
- Nomónica
- Equivalente azimuthal de Lambert
- Ortogonal
- Estereográfica polar
- Estereográfica

Parámetros de proyección de correlaciones

Tabla 64. Parámetros de proyección de correlaciones

Parámetro	Descripción
central_meridian	La línea de longitud elegida como origen de coordenadas x.
scale_factor	Se utiliza generalmente para reducir la cantidad de distorsión en una proyección de correlaciones.

Tabla 64. Parámetros de proyección de correlaciones (continuación)

Parámetro	Descripción
standard_parallel_1	Una línea de latitud que no tiene distorsión a nivel general. También se utiliza para ″latitud de escala real″.
standard_parallel_2	Una línea de latitud que no tiene distorsión a nivel general.
longitude_of_center	La longitud que define el punto central de la proyección de correlaciones.
latitude_of_center	La latitud que define el punto central de la proyección de correlaciones.
latitude_of_origin	La latitud elegida como el origen de las coordenadas y.
false_easting	Se suma a las coordenadas x. Sirve para ofrecer valores positivos.
false_northing	Se suma a las coordenadas y. Sirve para ofrecer valores positivos.
azimuth	El ángulo este de norte que define la línea central de una proyección oblicua.
longitude_of_point_1	La longitud del primer punto necesario para una proyección de correlaciones.
latitude_of_point_1	La latitud del primer punto necesario para una proyección de correlaciones.
longitude_of_point_2	La longitud del segundo punto necesario para una proyección de correlaciones.
latitude_of_point_2	La latitud del segundo punto necesario para una proyección de correlaciones.
longitude_of_point_3	La longitud del tercer punto necesario para una proyección de correlaciones.
latitude_of_point_3	La latitud del tercer punto necesario para una proyección de correlaciones.
landsat_number	El número de un satélite Landsat.
path_number	El número de ruta orbital de un determinado satélite.
perspective_point_height	La altura por encima de la tierra del punto de perspectiva de la proyección de correlaciones.
fipszone	Número de zona del Sistema de coordenadas de plano de estado.
zone	Número de zona UTM.

Capítulo 16. Formatos de archivos para datos espaciales

Este capítulo documenta las representaciones conocidas de DB2 Spatial Extender. Las representaciones se denominan *conocidas* porque las ofrece ESRI y no son específicas de DB2 Spatial Extender. Estos tres tipos de valores espaciales son importantes para comprender el proceso de importación y exportación de datos espaciales:

- Las representaciones de texto conocido de Open GIS Consortium (OGC)
- Las representaciones de binario conocido (WKB) de OGC
- Las representaciones de forma ESRI

Las representaciones de texto conocido OGC

DB2 Spatial Extender tiene varias funciones que generan geometrías a partir de descripciones de texto:

ST_GeomFromText
Crea una geometría a partir de una representación de texto de cualquier tipo de geometría.

ST_PointFromText
Crea un punto a partir de una representación de texto de punto.

ST_LineFromText
Crea una serie lineal a partir de una representación de texto de serie lineal.

ST_PolyFromText
Crea un polígono a partir de una representación de texto de polígono.

ST_MPointFromText
Crea una geometría varios puntos a partir de representaciones de texto de varios puntos.

ST_MLineFromText
Crea una geometría varias series lineales a partir de una representación de texto de varias series lineales.

ST_MPolyFromText
Crea una geometría varios polígonos a partir de una representación de texto de varios polígonos.

La representación de texto es una serie de formato de texto ASCII que permite intercambiar una geometría en formato de texto ASCII. Puede utilizar estas funciones en un programa en lenguaje de tercera o cuarta generación (3GL o

4GL) porque no necesitan definiciones de ninguna estructura de programa especial. La función ST_AsText convierte una geometría existente en una representación de texto.

Cada tipo de geometría tiene una representación de texto conocido, que se puede utilizar para crear nuevas instancias del tipo y para convertir las instancias existentes a formato textual para visualizarlas de forma alfanumérica.

La representación de texto conocido de una geometría se define del siguiente modo: la notación {}* indica 0 o más repeticiones de las señales entre llaves; las llaves no aparecen en la lista de señales de salida.

```
<Geometry Tagged Text> :=
| <Point Tagged Text>
| <LineString Tagged Text>
| <Polygon Tagged Text>
| <MultiPoint Tagged  Text>
| <MultiLineString Tagged Text>
| <MultiPolygon Tagged Text>

<Point Tagged Text> :=
POINT <Point Text>

<LineString Tagged Text> :=
LINESTRING <LineString Text>

<Polygon Tagged Text> :=
POLYGON <Polygon Text>

<MultiPoint Tagged Text> :=
MULTIPOINT <Multipoint Text>

<MultiLineString Tagged Text> :=
MULTILINESTRING <MultiLineString Text>

<MultiPolygon Tagged Text> :=
MULTIPOLYGON <MultiPolygon Text>

<Point Text> := EMPTY
|    <Point>
| Z  <PointZ>
| M  <PointM>
| ZM <PointZM>

<Point> :=  <x>  <y>
<x> := literal de doble precisión
<y> := literal de doble precisión
<PointZ> :=  <x>  <y>  <z>
<x> := literal de doble precisión
<y> := literal de doble precisión
<z> := literal de doble precisión
<PointM> :=  <x>  <y>  <m>
```

```
<x> := literal de doble precisión
<y> := literal de doble precisión
<m> := literal de doble precisión
<PointZM> :=  <x>  <y>  <z>  <m>
<x> := literal de doble precisión
<y> := literal de doble precisión
<z> := literal de doble precisión
<m> := literal de doble precisión

<LineString Text> := EMPTY
|    ( <Point Text >   {,  <Point Text> }*  )
| Z  ( <PointZ Text >  {,  <PointZ Text> }*  )
| M  ( <PointM Text >  {,  <PointM Text> }*  )
| ZM ( <PointZM Text > {,  <PointZM Text> }*  )

<Polygon Text> := EMPTY
| ( <LineString Text > {,< LineString Text > }*  )

<Multipoint Text> := EMPTY
| ( <Point Text >   {,  <Point Text > }*  )

<MultiLineString Text> := EMPTY
| ( <LineString Text > {,< LineString Text>}*  )

<MultiPolygon Text> := EMPTY
| ( < Polygon Text > {,  < Polygon Text > }*  )
```

La sintaxis de la función básica es:

```
función (<descripción texto>,<SRID>)
```

El SRID, el identificador de referencias espaciales y clave principal de la tabla SPATIAL_REFERENCES, identifica el sistema de referencias espaciales de la geometría que se almacena en la tabla SPATIAL_REFERENCES. Para que una geometría se pueda insertar en una columna espacial, su SRID debe coincidir con el SRID de la columna espacial.

La descripción de texto está formada por tres componentes básicos que se encierran entre comillas simples, por ejemplo:

```
<'tipo geometría'> ['tipo coordenadas'] [''lista coordenadas']
```

donde:

tipo geometría
: Es uno de los siguientes: punto, serie lineal, polígono, varios puntos, varias series lineales o varios polígonos.

tipo coordenadas
: Especifica si la geometría tiene o no coordenadas Z o medidas. Deje este argumento en blanco si la geometría no tiene ni coordenadas Z ni medidas. Si no es así, defina el tipo de coordenadas en Z para

geometrías que contienen coordenadas Z, M para geometrías con medidas y ZM para geometrías con ambas.

lista coordenadas
Define los vértices de la geometría. Las listas de coordenadas están delimitadas por comas y encerradas entre paréntesis. Las geometrías con varios componentes necesitan grupos de paréntesis para encerrar cada parte del componente. Si la geometría está vacía, la palabra clave EMPTY sustituye a la coordenada.

La Tabla 65 muestra una lista de ejemplos de todas las posibles representaciones de texto.

Tabla 65. Tipos de geometrías y sus representaciones de texto

Tipo de geometría	Descripción de texto	Comentario
punto	punto vacío	punto vacío
punto	punto z vacío	punto vacío con coordenada z
punto	punto m vacío	punto vacío con medida
punto	punto zm vacío	punto vacío con coordenada z y medida
punto	punto (10,05 10,28)	punto
punto	punto z (10,05 10,28 2,51)	punto con coordenada z
punto	punto m (10,05 10,28 4,72)	punto con medida
punto	punto zm (10,05 10,28 2,51 4,72)	punto con coordenada z y medida
serie lineal	serie lineal vacía	serie lineal vacía
serie lineal	serie lineal z vacía	serie lineal vacía con coordenadas z
serie lineal	serie lineal m vacía	serie lineal vacía con medidas
serie lineal	serie lineal zm vacía	serie lineal vacía con coordenadas z y medidas
serie lineal	serie lineal (10,05 10,28 , 20,95 20,89)	serie lineal
serie lineal	serie lineal z (10,05 10,28 3,09, 20,95 31,98 4,72, 21,98 29,80 3,51)	serie lineal con coordenadas z
serie lineal	serie lineal m (10,05 10,28 5,84, 20,95 31,98 9,01, 21,98 29,80 12,84)	serie lineal con medidas

Tabla 65. Tipos de geometrías y sus representaciones de texto (continuación)

Tipo de geometría	Descripción de texto	Comentario
serie lineal	serie lineal zm ( )	serie lineal con coordenadas z y medidas
polígono	polígono vacío	polígono vacío
polígono	polígono z vacío	polígono vacío con coordenadas z
polígono	polígono m vacío	polígono vacío con medidas
polígono	polígono zm vacío	polígono vacío con coordenadas z y medidas
polígono	polígono ((10 10, 10 20, 20 20, 20 15, 10 10))	polígono
polígono	polígono z (( ))	polígono con coordenadas z
polígono	polígono m (( ))	polígono con medidas
polígono	polígono zm (( ))	polígono con coordenadas z y medidas
varios puntos	varios puntos vacío	varios puntos vacío
varios puntos	varios puntos z vacío	varios puntos vacío con coordenadas z
varios puntos	varios puntos m vacío	varios puntos vacío con medidas
varios puntos	varios puntos zm vacío	varios puntos vacío con coordenadas z y medidas
varios puntos	varios puntos vacío	varios puntos vacío
varios puntos	varios puntos (10 10, 20 20)	varios puntos con dos puntos
varios puntos	varios puntos z (10 10 2, 20 20 3)	varios puntos con coordenadas z
varios puntos	varios puntos m (10 10 4, 20 20 5)	varios puntos con medidas
varios puntos	varios puntos zm (10 10 2 4, 20 20 3 5)	varios puntos con coordenadas z y medidas
varias series lineales	varias series lineales vacío	varias series lineales vacío
varias series lineales	varias series lineales z vacío	varias series lineales vacío con coordenadas z
varias series lineales	varias series lineales m vacío	varias series lineales con medidas
varias series lineales	varias series lineales zm vacío	varias series lineales con coordenadas z y medidas

Tabla 65. Tipos de geometrías y sus representaciones de texto (continuación)

Tipo de geometría	Descripción de texto	Comentario
varias series lineales	varias series lineales (( ))	varias series lineales
varias series lineales	varias series lineales z (( ))	varias series lineales con coordenadas z
varias series lineales	varias series lineales m (( ))	varias series lineales con medidas
varias series lineales	varias series lineales zm (( ))	varias series lineales con coordenadas z y medidas
varios polígonos	varios polígonos vacío	varios polígonos vacío
varios polígonos	varios polígonos z vacío	varios polígonos vacío con coordenadas z
varios polígonos	varios polígonos m vacío	varios polígonos vacío con medidas
varios polígonos	varios polígonos z	varios polígonos vacío con coordenadas z y medidas
varios polígonos	varios polígonos ((( )))	varios polígonos
varios polígonos	varios polígonos z ((( )))	varios polígonos con coordenadas z
varios polígonos	varios polígonos m (((10 10 2, 10 20 3, 20 20 4, 20 15 5, 10 10 2), (50 40 7, 50 50 3, 60 50 4, 60 40 5, 50 40 7)))	varios polígonos con medidas
varios polígonos	varios polígonos zm ((( )))	varios polígonos con coordenadas z y medidas

Las representaciones de binario conocido (WKB) de OGC

DB2 Spatial Extender tiene varias funciones que generan geometrías a partir de representaciones binarias:

ST_GeomFromWKB
Crea una geometría a partir de una representación WKB de cualquier tipo de geometría.

ST_PointFromWKB
Crea un punto a partir de una representación WKB de punto.

ST_LineFromWKB
Crea una serie lineal a partir de una representación WKB de serie lineal.

ST_PolyFromWKB
Crea un polígono a partir de una representación WKB de polígono.

ST_MPointFromWKB
Crea una geometría varios puntos a partir de una representación WKB de varios puntos.

ST_MLineFromWKB
Crea una geometría varias series lineales a partir de una representación WKB de varias series lineales.

ST_MPolyFromWKB
Crea una geometría varios polígonos a partir de una representación WKB de varios polígonos.

La representación de binario conocido es una corriente continua de bytes. Permite intercambiar una geometría entre un cliente ODBC y una base de datos SQL en formato binario. Puesto que estas funciones de geometría necesitan la definición de estructuras de lenguaje de programación C para correlacionar la representación binaria, están destinadas a ser utilizadas dentro de un programa de lenguaje de tercera generación (3GL). No se ajustan a un entorno de lenguaje de cuarta generación (4GL). La función ST_AsBinary convierte un valor de geometría existente en una representación de binario conocido.

La representación de binario conocido correspondiente a la geometría se obtiene serializando una instancia de la geometría como una secuencia de tipos numéricos. Estos tipos se extraen del grupo (entero sin signo, doble), y luego cada tipo numérico se serializa como una secuencia de bytes. Los tipos se serializan utilizando una de las dos representaciones de binario, estándar y bien definido para tipos numéricos (NDR, XDR). Un identificador de un byte que precede los bytes serializados describe la codificación específica de binario (NDR o XDR) utilizada para una corriente de bytes de geometría. La única diferencia entre las dos codificaciones de geometría es el orden de uno de los bytes: la codificación XDR es Big Endian; la codificación NDR es Little Endian.

Definiciones de tipos numéricos

Un *entero sin signo* es un tipo de datos de 32 bits (4 bytes) que codifica un entero no negativo dentro del rango [0, 4294967295].

Un *doble* es un tipo de datos de doble precisión de 64 bits (8 bytes) que codifica un número de doble precisión utilizando el formato de doble precisión IEEE 754.

Estas definiciones son comunes a XDR y a NDR.

Codificación XDR (Big Endian) de tipos numéricos

La representación XDR de un entero sin signo es Big Endian (primero el byte más significativo).

La representación XDR de un doble es Big Endian (el bit de signo es el primer byte).

Codificación NDR (Little Endian) de tipos numéricos

La representación NDR de un entero sin signo es Little Endian (primero el byte menos significativo).

La representación NDR de un doble es Little Endian (el bit de signo es el último byte).

Conversión entre NDR y XDR

La conversión entre los tipos de datos NDR y XDR para enteros sin signo y dobles es una operación sencilla. Se debe invertir el orden de los bytes dentro de cada entero sin signo o doble de la corriente de bytes.

Descripción de las corrientes de bytes WKBGeometry

Esta sección describe la representación de binario conocido correspondiente a una geometría. El componente básico es la corriente de bytes correspondiente a un punto, que consta de dos dobles. Las corrientes de bytes de otras geometrías se crean utilizando las corrientes de bytes correspondientes a geometrías que ya están definidas.

```
// Definiciones de tipo básico
// byte : 1 byte
// uint32 : entero sin signo de 32 bits (4 bytes)
// double : número de doble precisión (8 bytes)

// Componentes básicos : Point, LinearRing

Point {
  double x;
  double y;
};
LinearRing   {
  uint32  numPoints;
  Point   points[numPoints];
};
enum wkbGeometryType {
  wkbPoint = 1,
  wkbLineString = 2,
  wkbPolygon = 3,
  wkbMultiPoint = 4,
  wkbMultiLineString = 5,
  wkbMultiPolygon = 6,
};
enum wkbByteOrder {
  wkbXDR = 0,                                    // Big Endian
  wkbNDR = 1                                   // Little Endian
};
WKBPoint {
  byte     byteOrder;
  uint32   wkbType;                                      // 1
  Point    point;
```

```
};
WKBLineString {
  byte     byteOrder;
  uint32   wkbType;                                          // 2
  uint32   numPoints;
  Point    points[numPoints];
}

WKBPolygon    {
  byte                byteOrder;
  uint32            wkbType;                                 // 3
  uint32            numRings;
  LinearRing        rings[numRings];
}
WKBMultiPoint    {
  byte                byteOrder;
  uint32            wkbType;                                 // 4
  uint32            num_wkbPoints;
  WKBPoint            WKBPoints[num_wkbPoints];
}
WKBMultiLineString    {
  byte              byteOrder;
  uint32            wkbType;                                 // 5
  uint32            num_wkbLineStrings;
  WKBLineString     WKBLineStrings[num_wkbLineStrings];
}

wkbMultiPolygon {
  byte              byteOrder;
  uint32            wkbType;                                 // 6
  uint32            num_wkbPolygons;
  WKBPolygon        wkbPolygons[num_wkbPolygons];
}

WKBGeometry  {
  union {
    WKBPoint                point;
    WKBLineString           linestring;
    WKBPolygon              polygon;
    WKBMultiPoint           mpoint;
    WKBMultiLineString      mlinestring;
    WKBMultiPolygon         mpolygon;
  }
};
```

La siguiente figura muestra una representación NDR.

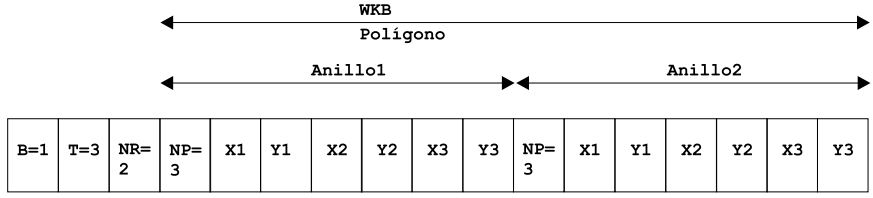


Figura 39. Representación en formato NDR. (B=1) de tipo polígono (T=3) con 2 lineales (NR=2), cada anillo tiene 3 puntos (NP=3).

Declaraciones sobre la representación WKB

La representación de binario conocido correspondiente a una geometría está diseñada para representar instancias de los tipos de geometrías descritos en el Modelo de objetos geometría y en la Especificación OpenGIS Abstract.

Estas declaraciones implican lo siguiente para anillos, polígonos y varios polígonos:

Anillos lineales
Los anillos son sencillos y cerrados, lo que significa que los anillos lineales no pueden formar intersección con ellos mismos.

Polígonos
En el límite de un polígono no puede haber dos anillos lineales que se crucen entre sí. Los anillos lineales del límite de un polígono pueden formar intersección como máximo en un punto, pero sólo como una tangente.

Varios polígonos
Los interiores de dos polígonos que son elementos de una geometría varios polígonos no pueden formar intersección. Los límites de dos polígonos que son elementos de una geometría varios polígonos pueden tocarse sólo en un número finito de puntos.

Las representaciones de forma ESRI

DB2 Spatial Extender tiene varias funciones que generan geometrías a partir de representaciones de forma ESRI. Además de la representación de dos dimensiones soportada por la representación de binario conocido Open GIS, la representación de forma ESRI también da soporte a coordenadas Z y medidas opcionales. Las siguientes funciones generan una geometría a partir de una forma ESRI:

ST_GeometryFromShape
Crea una geometría a partir de una representación de forma de cualquier tipo de geometría.

ST_PointFromShape
Crea un punto a partir de una representación de forma de punto.

ST_LineFromShape
Crea una serie lineal a partir de una representación de forma de serie lineal.

ST_PolyFromShape
Crea un polígono a partir de una representación de forma de polígono.

ST_MPointFromShape
Crea una geometría varios puntos a partir de una representación de forma de varios puntos.

ST_MLineFromShape
Crea una geometría varias series lineales a partir de una representación de forma de varias series lineales.

ST_MPolyFromShape
Crea una geometría varios polígonos a partir de una representación de forma de varios polígonos.

La sintaxis general de estas funciones es la misma. El primer argumento es la representación de forma que se entra como tipo de datos objeto grande binario (BLOB). El segundo argumento es el entero identificador de referencias espaciales que se va a asignar a la geometría. La función GeometryFromShape tiene la siguiente sintaxis:

```
db2gse.GeometryFromShape(ShapeGeometry Blob(1M), cr db2gse.coordref)
```

Puesto que estas funciones de forma necesitan la definición de estructuras de lenguaje de programación C para correlacionar la representación binaria, están destinadas a ser utilizadas dentro de un programa 3GL y no se ajustan a un entorno 4GL. La función AsShape convierte el valor de geometría en una representación de forma ESRI.

Un tipo de forma de 0 indica una forma nula, sin datos geométricos para la forma.

Valor	Shape Type
0	Forma nula
1	Point
3*	PolyLine
5	Polygon
8	MultiPoint
11	PointZ

Valor	Shape Type
13	PolyLineZ
15	PolygonZ
18	MultiPointZ
21	PointM
23	PolyLineM
25	PolygonM
28	MultiPointM

Nota: * Los tipos de formas no especificados en la tabla anterior (2, 4, 6, etc.) se reservan para un futuro uso.

Tipos de formas en espacio XY

Point

Un punto consta de un par de coordenadas de doble precisión en el orden X, Y.

Tabla 66. Contenido de la corriente de bytes de un punto

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Shape Type	1	Integer	1	Little
Byte 4	X	X	Double	1	Little
Byte 12	Y	Y	Double	1	Little

MultiPoint

Una geometría varios puntos consta de un grupo de puntos. El recuadro límite se almacena en el orden Xmín, Ymín, Xmáx, Ymáx.

Tabla 67. Contenido de la corriente de bytes de varios puntos

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Tipo de forma	8	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumPoints	NumPoints	Integer	1	Little
Byte 40	Points	Points	Point	NumPoints	Little

PolyLine

Una geometría PolyLine es un conjunto ordenado de vértices que consta de una o más partes. Una parte es una secuencia conectada de dos o más puntos. Los puntos pueden estar o no conectados entre sí. Las partes pueden formar o no formar intersección entre sí.

Puesto que esta especificación no prohíbe puntos consecutivos con coordenadas idénticas, los lectores de archivos de forma deben manejar estos casos. Por otro lado, no se permiten las partes degeneradas de longitud cero que pueden derivar de dichos casos.

Los campos de una PolyLine son:

Box El recuadro límite de la geometría PolyLine que se almacena en el orden Xmín, Ymín, Xmáx, Ymáx.

NumParts
El número de partes de una geometría PolyLine.

NumPoints
El número total de puntos correspondiente a todas las partes.

Parts Una matriz de longitud NumParts. Cada PolyLine almacena el índice de su primer punto en la matriz de puntos. Los índices de la matriz son con respecto a 0.

Points Una matriz de longitud NumPoints. Los puntos de cada parte de la geometría PolyLine se almacenan de extremo a extremo. Los puntos correspondientes a la parte 2 siguen a los correspondientes a la parte 1, y así sucesivamente. La matriz de partes alberga el índice de matriz del punto inicial correspondiente a cada parte. No hay ningún delimitador en la matriz de puntos entre partes.

Tabla 68. Contenido de la corriente de bytes de PolyLine

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Shape Type	3	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little

Nota: X = 44 + 4 * NumParts.

Polygon

Un polígono consta de uno o más anillos. Un anillo es una secuencia conectada de cuatro o más puntos que forman un bucle cerrado que no forma intersección consigo mismo. Un polígono puede contener varios anillos externos. El orden de vértices o la orientación de un anillo indica qué parte del anillo es el interior del polígono. La parte que queda a la derecha del observador que camina por el anillo en orden de vértices es la parte interna del polígono. Los vértices de los anillos que definen orificios en polígonos están en una dirección contraria a la de las agujas del reloj. Los vértices correspondientes a un solo polígono con anillos están, por tanto, siempre en el orden de las agujas del reloj. Los anillos de un polígono se denominan partes.

Puesto que esta especificación no prohíbe puntos consecutivos con coordenadas idénticas, los lectores de archivos de forma deben manejar estos casos. Por otro lado, no se permiten las partes degeneradas de longitud cero o de área cero que pueden derivar de dichos casos.

Los campos de un polígono son:

Box El recuadro límite del polígono que se almacena en el orden Xmín, Ymín, Xmáx, Ymáx.

NumParts
El número de anillos del polígono.

NumPoints
El número total de puntos correspondiente a todos los anillos.

Parts Una matriz de longitud NumParts. Almacena, para cada anillo, el índice de su primer punto en la matriz de puntos. Los índices de la matriz son con respecto a 0.

Points Una matriz de longitud NumPoints. Los puntos de cada anillo del polígono se almacenan de extremo a extremo. Los puntos correspondientes al anillo 2 siguen a los correspondientes al anillo 1, y así sucesivamente. La matriz de partes alberga el índice de matriz del punto inicial correspondiente a cada anillo. No hay ningún delimitador en la matriz de puntos entre anillos.

Notas importantes sobre las formas Polygon:

- Los anillos son cerrados (el primer y último vértice de un anillo DEBE ser el mismo).
- El orden de anillos en la matriz de puntos no es significativo.
- Los polígonos almacenados en un archivo de forma deben estar limpios. Un polígono limpio es uno que:
  - No forma intersección consigo mismo. Esto significa que un segmento que pertenece a un anillo no puede formar intersección con un segmento que pertenece a otro anillo. Los anillos de un

polígono se pueden tocar entre sí por los vértices, pero no a lo largo de los segmentos. Se considera que los segmentos colineales forman intersección.

– Tiene el interior del polígono en el lado ″correcto″ de la línea que lo define. La parte que queda a la derecha de un observador que camina por el anillo en orden de vértices es la parte interior del polígono. Los vértices correspondientes a un solo polígono con anillos están, por tanto, siempre en el orden de las agujas del reloj. Los anillos que definen orificios en estos polígonos tienen una orientación contraria a la de las agujas del reloj.

Los polígonos ″sucios″ son aquellos cuyos anillos que definen orificios en el polígono también van en el sentido de las agujas del reloj, lo que causa que los interiores se solapen.

Una instancia de polígono de ejemplo:

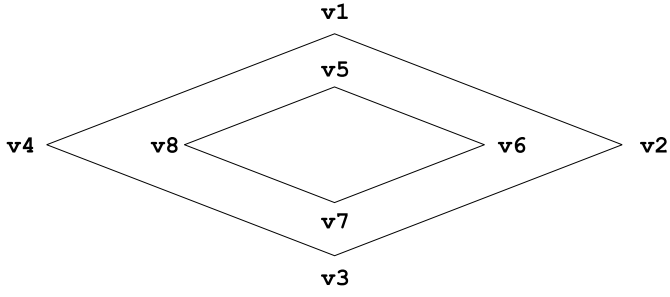


Figura 40. Un polígono con un orificio y ocho vértices

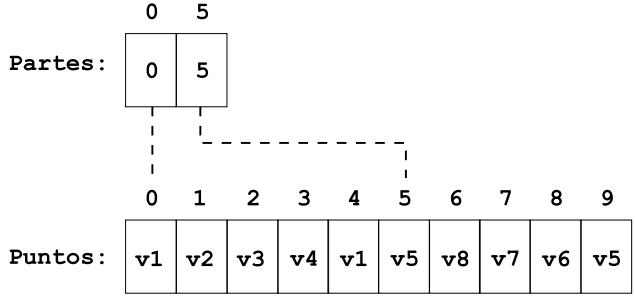


Figura 41. Contenido de la corriente de bytes del polígono. NumParts igual a 2 y NumPoints igual a 10. Observe que el orden de los puntos del polígono en forma de rosquilla (con orificio) está invertido.

Tabla 69. Contenido de la corriente de bytes de un polígono

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Shape Type	5	Integer	1	Little

Tabla 69. Contenido de la corriente de bytes de un polígono (continuación)

Posición	Campo	Valor	Tipo	Número	Orden
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little

Nota: X = 44 + 4 * NumParts.

Tipos de formas con medidas en espacio XY

PointM

Un PointM consta de un par de coordenadas de doble precisión en el orden X, Y, más una medida M.

Tabla 70. Contenido de la corriente de bytes de PointM

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Shape Type	21	Integer	1	Little
Byte 4	X	X	Double	1	Little
Byte 12	Y	Y	Double	1	Little
Byte 20	M	M	Double	1	Little

MultiPointM

Los campos de un MultiPointM son:

Box El recuadro límite del MultiPointM almacenado en el orden Xmín, Ymín, Xmáx, Ymáx.

NumPoints
El número de puntos.

Points Una matriz de puntos de longitud NumPoints.

NumMs
El número de medidas que siguen. NumMs sólo puede tener dos valores: cero si este campo no va seguido de ninguna medida, e igual a NumPoints si hay medidas.

M Range
Las medidas mínima y máxima correspondientes a MultiPointM almacenadas en el orden Mmín, Mmáx.

M Array
Una matriz de medidas de longitud NumPoints.

Tabla 71. Contenido de la corriente de bytes de MultiPointM

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Shape Type	28	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumPoints	NumPoints	Integer	1	Little
Byte 40	Points	Points	Point	NumPoints	Little
Byte X	NumMs	NumMs	Integer	1	Little
Byte X+4*	Mmin	Mmin	Double	1	Little
Byte X+12*	Mmax	Mmax	Double	1	Little
Byte X+20*	Marray	Marray	Double	NumPoints	Little

Notas:

1. X = 40 + (16 * NumPoints)
2. * opcional

PolyLineM

Un archivo de forma PolyLineM consta de una o más partes. Una parte es una secuencia conectada de dos o más puntos. Las partes pueden estar o no conectadas entre sí. Las partes pueden formar o no formar intersección entre sí.

Los campos de un PolyLineM son:

Box El recuadro límite del PolyLineM almacenado en el orden Xmín, Ymín, Xmáx, Ymáx.

NumParts
El número de partes del PolyLineM.

NumPoints
El número total de puntos correspondiente a todas las partes.

Parts Una matriz de longitud NumParts. Almacena, para cada parte, el índice de su primer punto en la matriz de puntos. Los índices de la matriz son con respecto a 0.

Points Una matriz de longitud NumPoints. Los puntos de cada parte de la geometría PolyLineM se almacenan de extremo a extremo. Los puntos correspondientes a la parte 2 siguen a los correspondientes a la parte 1, y así sucesivamente. La matriz de partes alberga el índice de matriz del punto inicial correspondiente a cada parte. No hay ningún delimitador en la matriz de puntos entre partes.

NumMs

El número de medidas que siguen. NumMs sólo puede tener dos valores: cero si este campo no va seguido de ninguna medida, e igual a NumPoints si hay medidas.

M Range

Las medidas mínima y máxima correspondientes a PolyLineM almacenadas en el orden Mmín, Mmáx.

M Array

Una matriz de longitud NumPoints. Las medidas de cada parte de la geometría PolyLineM se almacenan de extremo a extremo. Las medidas correspondientes a la parte 2 siguen a las correspondientes a la parte 1, y así sucesivamente. La matriz de partes alberga el índice de matriz del punto inicial correspondiente a cada parte. No hay ningún delimitador en la matriz de medidas entre partes.

Tabla 72. Contenido de la corriente de bytes de PolyLineM

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Shape Type	13	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y	NumMs	NumMs	Integer	1	Little
Byte Y+4*	Mmin	Mmin	Double	1	Little
Byte Y+12*	Mmax	Mmax	Double	1	Little
Byte Y+20*	Marray	Marray	Double	NumPoints	Little

Notas:

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints).
2. * opcional

PolygonM

Un PolygonM consta de un número de anillos. Un anillo es un bucle cerrado que no forma intersección consigo mismo. Observe que las intersecciones se calculan en espacio XY, *no* en espacio XYM. Un PolygonM puede contener varios anillos externos. Los anillos de un PolygonM se denominan partes.

Los campos de un PolygonM son:

Box El recuadro límite del PolygonM almacenado en el orden Xmín, Ymín, Xmáx, Ymáx.

NumParts
El número de anillos del PolygonM.

NumPoints
El número total de puntos correspondiente a todos los anillos.

Parts Una matriz de longitud NumParts. Almacena, para cada anillo, el índice de su primer punto en la matriz de puntos. Los índices de la matriz son con respecto a 0.

Points Una matriz de longitud NumPoints. Los puntos de cada anillo del PolygonM se almacenan de extremo a extremo. Los puntos correspondientes al anillo 2 siguen a los correspondientes al anillo 1, y así sucesivamente. La matriz de partes alberga el índice de matriz del punto inicial correspondiente a cada anillo. No hay ningún delimitador en la matriz de puntos entre anillos.

NumMs
El número de medidas que siguen. NumMs sólo puede tener dos valores: cero si este campo no va seguido de ninguna medida o igual a NumPoints si hay medidas.

M Range
Las medidas mínima y máxima correspondientes a PolygonM almacenadas en el orden Mmín, Mmáx.

M Array
Una matriz de longitud NumPoints. Las medidas de cada anillo del PolygonM se almacenan de extremo a extremo. Las medidas correspondientes al anillo 2 siguen a las correspondientes al anillo 1, y así sucesivamente. La matriz de partes alberga el índice de matriz de la medida inicial correspondiente a cada anillo. No hay ningún delimitador en la matriz de medidas entre anillos.

Notas importantes sobre las formas PolygonM:

- Los anillos son cerrados (el primer y el último vértice de un anillo debe ser el mismo).
- El orden de anillos en la matriz de puntos no es significativo.

Tabla 73. Contenido de la corriente de bytes de PolygonM

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Shape Type	15	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little

Tabla 73. Contenido de la corriente de bytes de PolygonM (continuación)

Posición	Campo	Valor	Tipo	Número	Orden
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y	NumMs	NumMs	Integer	1	Little
Byte Y+4*	Mmin	Mmin	Double	1	Little
Byte Y+12*	Mmax	Mmax	Double	1	Little
Byte Y+20*	Marray	Marray	Double	NumPoints	Little

Notas:

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints).
2. * opcional

Tipos de formas en espacio XYZ

PointZ

Un PointZ consta de tres coordenadas de doble precisión en el orden X, Y, Z más una medida.

Tabla 74. Contenido de la corriente de bytes de PointZ

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Shape Type	11	Integer	1	Little
Byte 4	X	X	Double	1	Little
Byte 12	Y	Y	Double	1	Little
Byte 20	Z	Z	Double	1	Little
Byte 28	Measure	M	Double	1	Little

MultiPointZ

Un MultiPointZ representa una serie de PointZ, del siguiente modo:

- El recuadro límite se almacena en el orden Xmín, Ymín, Xmáx, Ymáx.
- El rango de Z límite se almacena en el orden Zmín, Zmáx. El rango de M límite se almacena en el orden Mmín, Mmáx.

Tabla 75. Contenido de la corriente de bytes de MultiPointZ

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Shape Type	18	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumPoints	NumPoints	Integer	1	Little
Byte 40	Points	Points	Point	NumPoints	Little

Tabla 75. Contenido de la corriente de bytes de MultiPointZ (continuación)

Posición	Campo	Valor	Tipo	Número	Orden
Byte X	Zmin	Zmin	Double	1	Little
Byte X+8	Zmax	Zmax	Double	1	Little
Byte X+16	Zarray	Zarray	Double	NumPoints	Little
Byte Y	NumMs	NumMs	Integer	1	Little
Byte Y+4*	Mmin	Mmin	Double	1	Little
Byte Y+12*	Mmax	Mmax	Double	1	Little
Byte Y+20*	Marray	Marray	Double	NumPoints	Little

Notas:

1. X = 40 + (16 * NumPoints); Y = X + 16 + (8 * NumPoints)
2. * opcional

PolyLineZ

Un PolyLineZ consta de una o más partes. Una parte es una secuencia conectada de dos o más puntos. Las partes pueden estar o no conectadas entre sí. Las partes pueden formar o no formar intersección entre sí.

Los campos de un PolyLineZ son:

Box El recuadro límite del PolyLineZ almacenado en el orden Xmín, Ymín, Xmáx, Ymáx.

NumParts
El número de partes del PolyLineZ.

NumPoints
El número total de puntos correspondiente a todas las partes.

Parts Una matriz de longitud NumParts. Almacena, para cada parte, el índice de su primer punto en la matriz de puntos. Los índices de la matriz son con respecto a 0.

Points Una matriz de longitud NumPoints. Los puntos de cada parte de la geometría PolyLineZ se almacenan de extremo a extremo. Los puntos correspondientes a la parte 2 siguen a los correspondientes a la parte 1, y así sucesivamente. La matriz de partes alberga el índice de matriz del punto inicial correspondiente a cada parte. No hay ningún delimitador en la matriz de puntos entre partes.

Z Range
Los valores mínimo y máximo de Z correspondientes a PolyLineZ almacenados en el orden Zmín, Zmáx.

Z Array

Una matriz de longitud NumPoints. Los valores de Z para cada parte del PolyLineZ se almacenan de extremo a extremo. Los valores de Z correspondientes a la parte 2 siguen a los valores de Z correspondientes a la parte 1, y así sucesivamente. La matriz de partes alberga el índice de matriz del punto inicial correspondiente a cada parte. No hay ningún delimitador en la matriz Z entre partes.

NumMs

El número de medidas que siguen. NumMs sólo puede tener dos valores: cero si este campo no va seguido de ninguna medida, e igual a NumPoints si hay medidas.

M Range

Las medidas mínima y máxima correspondientes a PolyLineZ almacenadas en el orden Mmín, Mmáx.

M Array

Una matriz de longitud NumPoints. Las medidas para cada parte del PolyLineZ se almacenan de extremo a extremo. Las medidas correspondientes a la parte 2 siguen a las correspondientes a la parte 1, y así sucesivamente. La matriz de partes alberga el índice de matriz de la medida inicial correspondiente a cada parte. No hay ningún delimitador en la matriz de medidas entre partes.

Tabla 76. Contenido de la corriente de bytes de PolyLineZ

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Shape Type	13	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y	Zmin	Zmin	Double	1	Little
Byte Y+8	Zmax	Zmax	Double	1	Little
Byte Y+16	Zarray	Zarray	Double	NumPoints	Little
Byte Z	NumMs	NumMs	Integer	1	Little
Byte Z+4*	Mmin	Mmin	Double	1	Little
Byte Z+12*	Mmax	Mmax	Double	1	Little
Byte Z+20*	Marray	Marray	Double	NumPoints	Little

Notas:

1. X = 44 + (4 * NumParts), Y = X + (16 * NumPoints), Z = Y + 16 + (8 * NumPoints)
2. * opcional

PolygonZ

Un PolygonZ consta de un número de anillos. Un anillo es un bucle cerrado que no forma intersección consigo mismo. Un PolygonZ puede contener varios anillos externos. Los anillos de un PolygonZ se denominan partes.

Los campos de un PolygonM son:

Box El recuadro límite del PolygonZ almacenado en el orden Xmín, Ymín, Xmáx, Ymáx.

NumParts
El número de anillos del PolygonZ.

NumPoints
El número total de puntos correspondiente a todos los anillos.

Parts Una matriz de longitud NumParts. Almacena, para cada anillo, el índice de su primer punto en la matriz de puntos. Los índices de la matriz son con respecto a 0.

Points Una matriz de longitud NumPoints. Los puntos de cada anillo del PolygonZ se almacenan de extremo a extremo. Los puntos correspondientes al anillo 2 siguen a los correspondientes al anillo 1, y así sucesivamente. La matriz de partes alberga el índice de matriz del punto inicial correspondiente a cada anillo. No hay ningún delimitador en la matriz de puntos entre anillos.

Z Range
Los valores mínimo y máximo de Z correspondientes al arco almacenados en el orden Zmín, Zmáx.

Z Array
Una matriz de longitud NumPoints. Los valores Z de cada anillo del PolygonZ se almacenan de extremo a extremo. Los valores de Z correspondientes al anillo 2 siguen a los valores de Z correspondientes al anillo 1, y así sucesivamente. La matriz de partes alberga el índice de matriz del valor Z inicial correspondiente a cada anillo. No hay ningún delimitador en la matriz de valores Z entre anillos.

NumMs
El número de medidas que siguen. NumMs sólo puede tener dos valores: cero si este campo no va seguido de ninguna medida, e igual a NumPoints si hay medidas.

M Range

Las medidas mínima y máxima correspondientes a PolygonZ almacenadas en el orden Mmín, Mmáx.

M Array

Una matriz de longitud NumPoints. Las medidas de cada anillo del PolygonZ se almacenan de extremo a extremo. Las medidas correspondientes al anillo 2 siguen a las correspondientes al anillo 1, y así sucesivamente. La matriz de partes alberga el índice de matriz de la medida inicial correspondiente a cada anillo. No hay ningún delimitador en la matriz de medidas entre anillos.

Notas importantes sobre las formas PolygonZ:

- Los anillos son cerrados (el primer y último vértice de un anillo DEBE ser el mismo).
- El orden de anillos en la matriz de puntos no es significativo.

Tabla 77. Contenido de la corriente de bytes de PolygonZ

Posición	Campo	Valor	Tipo	Número	Orden
Byte 0	Shape Type	15	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y	Zmin	Zmin	Double	1	Little
Byte Y+8	Zmax	Zmax	Double	1	Little
Byte Y+16	Zarray	Zarray	Double	NumPoints	Little
Byte Z	NumMs	NumMs	Integer	1	Little
Byte Z+4*	Mmin	Mmin	Double	1	Little
Byte Z+12*	Mmax	Mmax	Double	1	Little
Byte Z+20*	Marray	Marray	Double	NumPoints	Little

Parte 3. Apéndices

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 geográfica. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se puede utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja 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
U.S.A.

En el caso de consultas sobre licencias referentes a información de doble byte (DBCS), consulte al Departamento de Propiedad Intelectual de IBM en su país o envíe consultas por escrito a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país en el que 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, COMERCIABILIDAD 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/o cambios en los productos y/o 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 esos sitios Web.

Cuando envía información a IBM, IBM puede utilizar o distribuir dicha información en la forma en que IBM 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 este) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

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 manual y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Convenio del Cliente IBM, el Convenio Internacional de Licencia de Programas de IBM o cualquier convenio equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse hecho 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 cualquier otra afirmación referente a productos no IBM. Las preguntas sobre las prestaciones de productos no 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.

Esta publicación 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 nombre 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 no intencionada.

LICENCIA DE COPYRIGHT:

Este manual puede contener programas de aplicación de ejemplo escritos en lenguaje fuente, que muestra técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de la forma que desee, sin pago alguno a IBM, con los fines de desarrollar, utilizar, comercializar o distribuir programas de aplicación 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 tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas.

Cada copia o porción 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 derivan de programas de ejemplo de IBM Corp. © Copyright IBM Corp. _especifique el año o años _. Reservados todos los derechos.

Marcas registradas

Los términos siguientes, que pueden estar indicados por un asterisco (*), son marcas registradas de International Business Machines Corporation en los Estados Unidos y/o en otros países.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

Los términos siguientes son marcas registradas de otras empresas:

Microsoft, Windows y Windows NT son marcas registradas de Microsoft Corporation.

Java, y las marcas registradas y logotipos basados en Java y Solaris son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos y/o en otros países.

Tivoli y NetView son marcas registradas de Tivoli Systems Inc. en los Estados Unidos y/o en otros países.

UNIX es una marca registrada en los Estados Unidos y/o en otros países bajo licencia exclusiva de X/Open Company Limited.

Otros nombres de empresas, productos o servicios, que pueden estar indicados por un doble asterisco (**), pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

G

H

I

J

L

M

P

R

S

T

U

V

W

X

Y

Z

Cómo ponerse en contacto con IBM

Si tiene un problema técnico, repase y lleve a cabo las acciones que se sugieren en la *Guía de resolución de problemas* antes de ponerse en contacto con el Centro de Asistencia al Cliente de DB2. Dicha guía sugiere información que puede reunir para ayudar al Centro de Asistencia a proporcionarle un mejor servicio.

Para obtener información o para solicitar cualquiera de los productos de DB2 Universal Database, consulte a un representante de IBM de una sucursal local o a un concesionario autorizado de IBM.

Si vive en los Estados Unidos, puede llamar a uno de los números siguientes:

- 1-800-237-5511 para obtener soporte técnico
- 1-888-426-4343 para obtener información sobre las opciones de servicio técnico disponibles

Información sobre productos

Si vive en los Estados Unidos, puede llamar a uno de los números siguientes:

- 1-800-IBM-CALL (1-800-426-2255) o 1-800-3IBM-OS2 (1-800-342-6672) para solicitar productos u obtener información general.
- 1-800-879-2755 para solicitar publicaciones.

http://www.ibm.com/software/data/
Las páginas Web de DB2 ofrecen información actual sobre DB2 referente a novedades, descripciones de productos, planes de formación, etc.

http://www.ibm.com/software/data/db2/library/
La biblioteca técnica de servicio y de productos DB2 ofrece acceso a las preguntas más frecuentes (FAQ), arreglos de programa, manuales e información técnica actualizada sobre DB2.

Nota: Puede que esta información sólo esté disponible en inglés.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
El sitio Web para el pedido de publicaciones internacionales proporciona información sobre cómo hacer pedidos de manuales.

http://www.ibm.com/education/certify/
El Programa de homologación profesional contenido en el sitio Web de IBM proporciona información de prueba de homologación para diversos productos de IBM, incluido DB2.

ftp.software.ibm.com

Conéctese como anónimo (anonymous). En el directorio /ps/products/db2 encontrará programas de demostración, arreglos de programa, información y herramientas referentes a DB2 y a muchos otros productos.

comp.databases.ibm-db2, bit.listserv.db2-l

En estos foros de discusión de Internet los usuarios pueden explicar sus experiencias con los productos DB2.

En Compuserve: GO IBMDB2

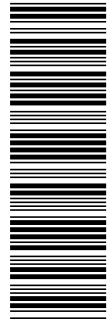
Entre este mandato para acceder a los foros referentes a la familia de productos DB2. Todos los productos DB2 tienen soporte a través de estos foros.

Para conocer cómo ponerse en contacto con IBM desde fuera de los Estados Unidos, consulte el Apéndice A del manual *IBM Software Support Handbook*. Para acceder a este documento, vaya a la página Web siguiente: http://www.ibm.com/support/ y luego seleccione el enlace ″IBM Software Support Handbook″, cerca del final de la página.

Nota: En algunos países, los distribuidores autorizados de IBM deben ponerse en contacto con su organización de soporte en lugar de acudir al Centro de Asistencia de IBM.

Número Pieza: CT7C0ES

Printed in Ireland



CT7C0ES

SC10-3528-00