IBM® DB2® Spatial Extender y
Geodetic Extender

Guía del usuario y manual de consulta

Versión 8.2

SC10-3755-01

IBM® DB2® Spatial Extender y
Geodetic Extender

Guía del usuario y manual de consulta

Versión 8.2

SC10-3755-01

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

Este manual es la traducción del original inglés *IBM DB2 Spatial Extender and Geodetic Extender User's Guide and Reference Version 8.2*, (SC27-1226-01).

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

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

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

Contenido

Tabla de contenido

Tabla de contenido

Parte 1. Información preliminar

Capítulo 1. Acerca de DB2 Spatial Extender

Este capítulo proporciona información preliminar sobre DB2 Spatial Extender; describe su función, los datos a los que da soporte y la forma en que están relacionados entre sí los conceptos básicos subyacentes.

La finalidad de DB2 Spatial Extender

Utilice DB2® Spatial Extender para generar y analizar información espacial con relación a elementos geográficos, y para almacenar y gestionar los datos en los que se basa esta información. Un *elemento geográfico* (algunas veces llamado *elemento* en esta explicación, para abreviar) es cualquier cosa del mundo real que tenga una ubicación identificable, o cualquier cosa que se pueda imaginar como existente en una ubicación identificable. Un elemento puede ser:

- Un objeto (es decir, una entidad concreta de cualquier tipo); por ejemplo, un río, un bosque o una cordillera de montañas.
- Un espacio; por ejemplo, una zona de seguridad alrededor de un lugar peligroso, o el área de mercado a la que da servicio un negocio en particular.
- Un suceso que se produce en una ubicación definible; por ejemplo, un accidente de circulación que se produce en un cruce de carreteras determinado, o una transacción de ventas en una tienda específica.

Los elementos existen en varios entornos. Por ejemplo, los objetos mencionados en la lista anterior—río, bosque, cordillera de montañas—pertenecen al entorno natural. Otros objetos, tales como ciudades, edificios y oficinas, pertenecen al entorno cultural. Incluso otros, tales como parques, zoológicos y granjas, representan una combinación de los entornos natural y cultural.

En esta explicación, el término *información espacial* hace referencia a la clase de información que DB2 Spatial Extender pone a disposición de sus usuarios, especialmente hechos y figuras con relación a las ubicaciones de elementos geográficos. Algunos ejemplos de información espacial son:

- Ubicaciones de elementos geográficos en el mapa (por ejemplo, los valores de longitud y latitud que definen dónde están situadas las ciudades)
- La ubicación de elementos geográficos con respecto a otros de ellos (por ejemplo, puntos dentro de una ciudad donde se encuentran hospitales y clínicas o la proximidad de las zonas residenciales de una ciudad con respecto a las zonas sísmicas locales)
- Modos en que los elementos geográficos se relacionan entre sí (por ejemplo, información sobre que un determinado sistema fluvial queda enclavado en una determinada región, o sobre que ciertos puentes de dicha región cruzan los afluentes del sistema fluvial)
- Las medidas que se utilizan para uno o varios elementos geográficos (por ejemplo, la distancia entre un edificio de oficinas y el límite del terreno o la longitud del perímetro de una reserva de aves)

La información espacial, ella sola o en combinación con datos relacionales tradicionales, puede ayudar al usuario en actividades como, por ejemplo, la definición de áreas en las que el usuario proporciona servicios, y en la determinación de ubicaciones de posibles mercados. 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 en que se proporciona servicio 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 estas ciudades 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 y DB2 puede producir información para responder a estas preguntas. Además, las herramientas frontales, aunque no son necesarias, pueden contribuir. Para ilustrarlo: una herramienta de visualización puede poner información producida por DB2 Spatial Extender—por ejemplo, la ubicación de concentraciones de clientes y la proximidad de carreteras principales a los restaurantes propuestos—en forma gráfica sobre un mapa. Las herramientas de Business Intelligence pueden poner información asociada—por ejemplo, nombres y descripciones de restaurantes de la competencia—en forma de informe.

Datos espaciales y geodésicos

Esta sección contiene una visión general de los datos que el usuario puede generar, almacenar y manejar para obtener información espacial. Los temas que abarca son los siguientes:

- Cómo los datos representan elementos geográficos
- La naturaleza de los datos espaciales y los datos geodésicos
- Modos de generar datos espaciales

Cómo los datos representan elementos geográficos

En DB2® Spatial Extender, un elemento geográfico se puede representar mediante uno o varios datos; por ejemplo, los datos de una fila de una tabla. (Un *dato elemental* es el valor o valores que ocupan una celda de una tabla relacional.) Por ejemplo, consideremos los 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. De la misma forma, cada fila de la tabla CUSTOMERS de la Figura 1 en la página 5, considerada como una unidad, representa un cliente del banco. Sin embargo, un subconjunto de cada una de las filas—específicamente, los datos que constituyen una 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. Datos que representan elementos geográficos. La fila de datos de la tabla BRANCHES representa una sucursal de un banco. Los datos de direcciones de la tabla CUSTOMERS representan la residencia de un cliente. Los nombres y las 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. En esta explicación se hace referencia a tales datos como *datos comerciales.*

Un subconjunto de los datos comerciales—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, USA. La dirección de un cliente es 9 Concourt Circle, San Jose, CA 95141, USA. DB2 Spatial Extender puede convertir estas direcciones en valores que indiquen dónde están situadas la sucursal y la residencia del cliente con respecto a otras direcciones. 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.

Puesto que la información espacial se obtendrá de los datos almacenados en la columna LOCATION, en esta explicación se hace referencia a estos datos como *datos espaciales.*

La naturaleza de los datos espaciales

Los datos espaciales están formados por coordenadas. Una *coordenada* es un número que indica una de las dos cosas siguientes:

- Una posición a lo largo de un eje relativo a un origen, dada una unidad de longitud.
- Una dirección relativa a una línea o un plano básicos, dada una medida angular.

Por ejemplo, la latitud es una coordenada que indica un ángulo relativo al plano ecuatorial y se expresa normalmente en grados. La longitud es una coordenada que indica un ángulo relativo al meridiano de Greenwich y también se expresa normalmente en grados. De este modo, en un mapa, la posición del Parque Nacional de Yellowstone está definida por la latitud de 44,45 grados al norte del ecuador y la longitud de 110,40 grados al oeste del meridiano de Greenwich. Y más precisamente, estas coordenadas se refieren al centro del Parque Nacional de Yellowstone en Estados Unidos.

Las definiciones de latitud y longitud, sus puntos, líneas y planos de referencia, unidades de medida y otros parámetros asociados reciben en conjunto el nombre de *sistema de coordenadas*. Los sistemas de coordenadas pueden basarse en valores distintos de la latitud y la longitud. Estos sistemas de coordenadas tienen sus propios puntos, líneas y planos de referencia, unidades de medida y parámetros asociados adicionales (como la transformación de proyección). Para obtener más información, consulte el apartado "Sistemas de coordenadas" en la página 59.

Los datos espaciales más simples consisten de un único par de coordenadas que define la posición de una única ubicación geográfica. Un dato espacial más amplio consta de varias coordenadas que definen un recorrido lineal, tal como el que formado por un camino o un río. Un tercer tipo consta de coordenadas que definen el perímetro de un área; por ejemplo, los límites de una parcela de tierra o de una zona con riesgo de inundación.

Cada dato espacial es un ejemplo de un tipo de dato espacial. El tipo de datos para coordenadas que marcan una ubicación determinada es ST_Point; el tipo de datos para coordenadas que definen un recorrido lineal es ST_LineString; y el tipo de datos para coordenadas que definen el perímetro de un área es ST_Polygon. Estos tipos, junto con los demás tipos de datos espaciales, son tipos estructurados que pertenecen a un jerarquía individual.

La naturaleza de los datos geodésicos

DB2 Geodetic Extender utiliza los mismos tipos de datos y funciones que Spatial Extender para almacenar datos geográficos en una base de datos de DB2. Los datos geodésicos son datos espaciales que se expresan en coordenadas de latitud y longitud, en un sistema coordenadas (consulte el apartado "Sistemas de coordenadas" en la página 59) que describe una superficie cerrada, continua y redonda. A diferencia de Spatial Extender, que considera que la tierra es un mapa plano, Geodetic Extender la considera un globo sin bordes ni costuras en los polos o en las líneas de datos. Un mapa plano necesita coordenadas proyectadas para transformar coordenadas esféricas en coordenadas planas, mientras que Geodetic Extender utiliza la latitud y la longitud en un modelo elipsoidal de la superficie de la tierra. Determinados cálculos como la intersección de líneas, el solapamiento de áreas, la distancia y el área son precisos, independientemente de la ubicación.

Para obtener más información, consulte los apartados "Cuándo utilizar DB2 Geodetic Extender y DB2 Spatial Extender" en la página 166 y "Diferencias de utilización entre las representaciones terrestres planas y redondas" en la página 205.

De dónde proceden los datos espaciales

Puede obtener datos espaciales:

- A partir de datos comerciales
- A partir de funciones espaciales
- Mediante importación desde fuentes externas

Utilización de datos comerciales como datos fuente

DB2 Spatial Extender puede obtener datos espaciales a partir de datos comerciales, tales como direcciones (tal como se menciona en el apartado "Cómo los datos representan elementos geográficos" en la página 4). Este proceso se denomina *geocodificación*. Para comprender la secuencia de procesos que intervienen, considere que la Figura 2 en la página 5 es una imagen "anterior" y que la Figura 3 es una imagen "posterior". La Figura 2 en la página 5 muestra que las tablas BRANCHES y CUSTOMERS tienen una columna designada para datos espaciales. Supongamos que DB2 Spatial Extender geocodifica las direcciones contenidas en estas tablas para obtener las coordenadas correspondientes a las direcciones y coloca las coordenadas en las columnas. La Figura 3 muestra 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 se han obtenido a partir de la dirección de las columnas ADDRESS, CITY, POSTAL CODE, STATE_PROV y COUNTRY. De forma similar, la columna LOCATION de la tabla BRANCHES contiene coordenadas que se han obtenido a partir de la dirección de las columnas ADDRESS, CITY, POSTAL CODE, STATE_PROV y COUNTRY de esta tabla.

DB2 Spatial Extender utiliza una función denominada *geocodificador* para convertir datos comerciales en coordenadas que permitan a las funciones trabajar con los datos.

Utilización de funciones para generar datos espaciales

Los datos espaciales se pueden generar no solamente mediante geocodificadores, sino también mediante otras funciones. Algunas de esas funciones, denominadas *constructores*, pueden generar datos espaciales a partir de valores que el usuario puede proporcionar como entrada. Otras requieren que existan datos espaciales como entrada. 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. Antes de que el banco pueda obtener esta información de la base de datos, necesita definir la zona que queda dentro de un radio especificado 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

delimitan los perímetros de las zonas. 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 varias funciones para obtener nuevos datos espaciales a partir de datos espaciales existentes. Para obtener más información acerca de estas funciones, consulte los apartados "Conceptos relacionados" y "Referencias relacionadas" al final de este tema.

Importación de datos espaciales

Un tercer modo de obtener datos espaciales consiste en importarlos a partir de archivos proporcionados por fuentes de datos externas. Estos archivos contienen habitualmente datos que se suelen aplicar a mapas: redes de calles, zonas con riesgo de inundación, fallas tectónicas, etcétera. La utilización de dichos datos en combinación con datos espaciales generados por el usuario, puede aumentar la información espacial 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 entonces importar datos sobre zonas con riesgo de inundación y sobre fallas tectónicas para ver cuáles de estas áreas conflictivas quedan comprendidas dentro de esta zona.

Conceptos relacionados:

- "DB2 Geodetic Extender" en la página 165
- "Cuándo utilizar DB2 Geodetic Extender y DB2 Spatial Extender" en la página 166
- "Funciones espaciales" en la página 307
- "Sistemas de coordenadas" en la página 59
- "Funciones que crean nuevas geometrías con configuraciones espaciales diferentes" en la página 337

Información relacionada:

- "Diferencias de utilización entre las representaciones terrestres planas y redondas" en la página 205
- "Funciones espaciales soportadas por DB2 Geodetic Extender" en la página 215
- "ST_Buffer" en la página 366
- "Funciones que generan nuevas geometrías a partir de geometrías existentes" en la página 336

Cómo están interrelacionados los elementos geográficos, la información espacial, los datos espaciales y las geometrías

Esta sección describe varios conceptos básicos sobre los que están basados las operaciones de DB2® Spatial Extender: los elementos geográficos, la información espacial, los datos espaciales y las geometrías.

DB2 Spatial Extender permite al usuario obtener datos y cifras referentes a entidades que se pueden definir en términos geográficos — es decir, su posición en la superficie terrestre o dentro de una región geográfica. La documentación de DB2 denomina a tales datos y cifras *información espacial*, y las entidades son los *elementos geográficos* (llamados simplemente *elementos* para abreviar).

Por ejemplo, podría utilizar DB2 Spatial Extender para determinar si una zona habitada queda dentro de una zona de vertidos propuesta. Las zonas habitadas y la zona de vertidos propuesta son elementos geográficos. El determinar si existe algún solapamiento entre ambas zonas sería un ejemplo de información espacial. Si se detecta que existe un solapamiento, su extensión sería también un ejemplo de información espacial.

Para producir información espacial, DB2 Spatial Extender debe procesar datos que definen las ubicaciones de los elementos geográficos. Estos datos, llamados *datos espaciales*, constan de coordenadas que indican las ubicaciones sobre una correlación o proyección similar. Por ejemplo, para determinar si un elemento geográfico se solapa con otro, DB2 Spatial Extender debe determinar dónde están situadas las coordenadas de uno de los elementos con respecto a las coordenadas del otro.

Dentro de la tecnología de la información espacial, los elementos geográficos se suelen representar mediante símbolos llamados *geometrías*. Las geometrías son entidades parcialmente visuales y parcialmente matemáticas. Considere su aspecto visual. El símbolo correspondiente a un elemento geográfico que tiene una anchura y extensión, tal como un parque o ciudad, es una figura de múltiples lados. Esta geometría se denomina *polígono*. El símbolo para un elemento lineal, tal como un río o una carretera, es una línea. Esta geometría se denomina *cadena lineal*.

La geometría representativa de un elemento geográfico tiene propiedades que se corresponden con datos referentes al elemento. La mayoría de estas propiedades se pueden expresar matemáticamente. Por ejemplo, las coordenadas de un elemento geográfico forman colectivamente una de las propiedades de la geometría correspondiente del elemento. Otra propiedad, denominada *dimensión*, es un valor numérico que indica si un elemento geográfico tiene una longitud o extensión.

Los datos espaciales y determinada información espacial se pueden considerar en términos de geometrías. Considere el ejemplo, descrito anteriormente, de las zonas habitadas y la zona de vertidos propuesta. Los datos espaciales de las zonas habitadas comprenden coordenadas que están guardadas en una columna de una tabla de una base de datos DB2. Por convenio, se considera que los datos almacenados no son simplemente datos, sino verdaderas geometrías. Debido a que las zonas habitadas tienen un ancho y una extensión, estas geometrías se pueden representar por polígonos.

Al igual que los datos espaciales, determinada información espacial se puede también considerar en términos de geometrías. Por ejemplo, para determinar si una zona habitada se solapa con una zona de vertidos propuesta, DB2 Spatial Extender

debe comparar las coordenadas del polígono representativo del vertedero con las coordenadas de los polígonos representativos de las zonas habitadas. La información resultante — es decir, las zonas de solapamiento — se considera que son polígonos: geometrías que tienen coordenadas, dimensión y otras propiedades.

Capítulo 2. Acerca de las geometrías

Este capítulo describe entidades de información, llamadas geometrías, que constan de coordenadas y representan elementos geográficos. Los temas que abarca son los siguientes:

- Geometrías
- Propiedades de las geometrías

Geometrías

Según el diccionario Webster de la lengua inglesa, la *geometría* es "La rama de las matemáticas que investiga las relaciones, propiedades y medida de los cuerpos sólidos, superficies, líneas y ángulos; la ciencia que trata de las propiedades y relaciones de las magnitudes; la ciencia de las relaciones del espacio". La palabra *geometría* también se ha utilizado para denotar las funciones geométricas utilizadas durante más de un siglo por los cartógrafos para trazar mapas del mundo. Según esta nueva acepción, una definición abstracta de geometría sería: "punto o agregado de puntos que representa un elemento geográfico de la tierra."

En DB2® Spatial Extender, la definición *práctica* de geometría es: "modelo de un elemento geográfico." El modelo se puede expresar en términos de las coordenadas del elemento geográfico. El modelo expresa información: por ejemplo, las coordenadas identifican la posición del elemento geográfico con respecto a puntos fijos de referencia. Además, el modelo se puede utilizar para generar información; por ejemplo, la función ST_Overlaps puede utilizar como datos de entrada las coordenadas de dos regiones próximas y notificar como información de salida si las regiones se solapan o no.

Las coordenadas del elemento geográfico representado por una geometría se consideran que son *propiedades* de la geometría. Algunos tipos de geometrías tienen también otras propiedades; por ejemplo, área, longitud y perímetro.

Las geometrías soportadas por DB2 Spatial Extender forman una jerarquía, que se muestra en la figura siguiente. La jerarquía de la geometría está definida por el documento de OpenGIS Consortium, Inc. (OGC) denominado "OpenGIS Simple Features Specification for SQL". De siete miembros de la jerarquía pueden crearse instancias. Es decir, se pueden definir con valores de coordenadas específicos y representar visualmente tal como se muestra en la figura.

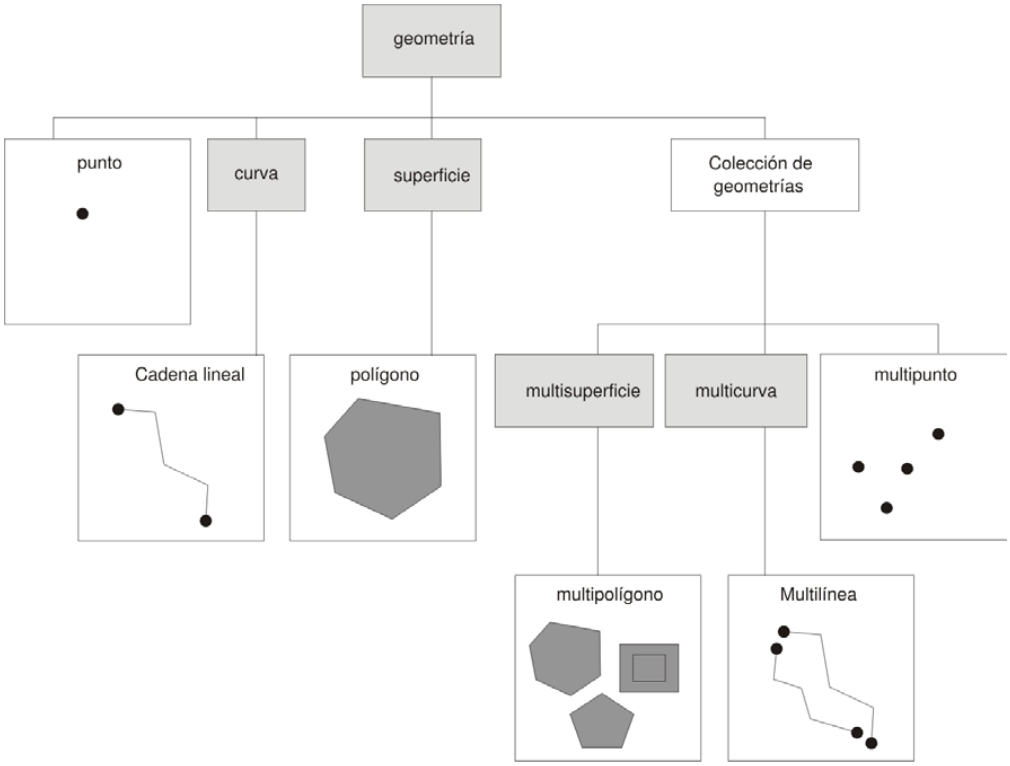


Figura 5. Jerarquía de geometrías soportadas por DB2 Spatial Extender. Las geometrías de esta figura de las que no pueden crearse instancias incluyen ejemplos de cómo se pueden representar visualmente.

Los tipos de datos espaciales soportados por DB2 Spatial Extender son implementaciones de las geometrías mostradas en la figura.

Tal como indica la figura, la superclase denominada *geometría* es la raíz de la jerarquía. Del tipo raíz y de otros subtipos adecuados de la jerarquía no pueden crearse instancias. Además, los usuarios pueden definir sus propios subtipos adecuados susceptibles o no de crear instancias.

Los subtipos se dividen en dos categorías: los subtipos de geometrías base y los subtipos de colección homogénea.

Las geometrías base comprenden:

Puntos
Un único punto. Los puntos representan elementos discretos que se considera que ocupan el lugar donde se cruzan una línea de coordenadas este-oeste (como un paralelo) y una línea de coordenadas norte-sur (como un meridiano). Por ejemplo, considere un mapa del mundo donde la posición de cada ciudad está representada por la intersección de un paralelo y un meridiano. Un punto podría representar cada ciudad.

Cadenas lineales
Una línea entre dos puntos. No tiene por qué ser una línea recta. Las cadenas lineales representan elementos geográficos lineales (por ejemplo, calles, canales y conductos).

Polígonos
Un polígono o una superficie dentro de un polígono. Los polígonos

representan elementos geográficos de múltiples lados (tales como barrios de viviendas sociales, bosques y hábitats salvajes).

Las colecciones homogéneas comprenden:

Multipuntos
Una colección de geometrías de puntos múltiples. Los multipuntos representan elementos geográficos que constan de varias partes cuyos componentes están situados 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 están situados en la intersección de un paralelo y un meridiano).

Multilíneas
Una colección de geometrías de curvas múltiples con múltiples cadenas. Las multilíneas representan elementos geográficos que constan de varias partes (por ejemplo, sistemas de ríos y sistemas de carreteras).

Multipolígonos
Una colección de geometrías de superficies múltiples con múltiples polígonos. Los multipolígonos representan elementos que constan de varias partes y están formados por unidades con varios lados (por ejemplo, terrenos agrícolas colectivos de una región determinada o un sistema de lagos).

Tal como sus nombres indican, las colecciones homogéneas son colecciones de geometrías base. Además de tener las propiedades de las geometrías base, las colecciones homogéneas tienen también algunas propiedades propias.

Conceptos relacionados:
• "Datos espaciales y geodésicos" en la página 4

Propiedades de las geometrías

Este tema describe las propiedades de las geometrías. Estas propiedades son:
- El tipo al que pertenece la geometría
- Las coordenadas de la geometría
- El interior, perímetro y exterior de la geometría
- La cualidad de ser simple o no simple
- La cualidad de estar vacía o no vacía
- El rectángulo delimitador mínimo de la geometría, también llamado envoltura
- La dimensión
- Identificador de sistemas de referencia espacial que está asociado a esta geometría

Tipo

Cada geometría pertenece a un tipo de la jerarquía de geometrías soportadas por DB2 Spatial Extender. Para obtener una descripción de la jerarquía de geometrías, consulte el apartado "Geometrías" en la página 11. Siete tipos de la jerarquía pueden definirse con valores de coordenadas específicos: puntos, cadenas lineales, polígonos, colecciones de geometrías, multipuntos, multilíneas y multipolígonos.

Coordenadas de una geometría

Todas las geometrías incluyen como mínimo una coordenada X y una coordenada Y, a menos que sean geometrías vacías, en cuyo caso no contendrán coordenadas de ningún tipo. Además, una geometría puede incluir una o más coordenadas Z y coordenadas M. Las coordenadas X, Y, Z y M se representan como números de precisión doble. En los siguientes subapartados se tratan estos temas:

- Coordenadas X e Y
- Coordenadas Z
- Coordenadas M

Coordenadas X e Y

Un *valor de coordenada X* indica una posición con respecto a un punto de referencia situado al este u oeste. Un *valor de coordenada Y* indica una posición con respecto a un punto de referencia situado al norte o sur.

Coordenadas Z

Algunas geometrías tienen asociado un valor de altitud o profundidad. Cada uno de los puntos que forman la geometría de un elemento geográfico puede incluir una coordenada Z opcional que representa una altitud o profundidad perpendicular a la superficie terrestre.

Coordenadas M

Una coordenada M (medida) es un valor que expresa información sobre un elemento geográfico y que se guarda junto con las coordenadas que definen la posición del elemento geográfico. Por ejemplo, suponga que está representando autopistas en su aplicación. Si desea que su aplicación procese valores que indican distancias lineales, puede guardar estos valores junto con las coordenadas que definen posiciones a lo largo de la autopista. Las coordenadas M se representan como números de precisión doble.

Interior, perímetro y exterior

Todas las geometrías ocupan una posición en el espacio definido por sus interiores, perímetro y exteriores. El exterior de una geometría es todo el espacio no ocupado por la geometría. El perímetro de una geometría actúa de límite entre el interior y el exterior de la geometría. El interior es el espacio ocupado por la geometría.

Simple o no simple

Los valores de algunos subtipos de geometría (cadenas lineales, multipuntos y multilíneas) pueden ser simples o no simples. Una geometría es simple si cumple todas las reglas topológicas impuestas sobre su subtipo, y es no simple en caso contrario. Una cadena lineal es simple si no forma intersección con su interior. Un multipunto es simple si ninguno de sus elementos ocupa el mismo espacio de coordenadas. Los puntos, las superficies y las multisuperficies son siempre simples.

Cerrada

Una curva está cerrada si sus puntos inicial y final son los mismos. Una multicurva está cerrada si cada uno de sus elementos está cerrado. Un anillo es una curva simple cerrada.

Vacía o no vacía

Una geometría está vacía si no tiene ningún punto. La envoltura, el perímetro, el interior y el exterior de una geometría vacía no están definidos y se representan como valores nulos. Una geometría vacía es siempre simple. Los polígonos y multipolígonos vacíos tienen un área igual a 0.

Rectángulo delimitador mínimo (MBR)

El MBR de una geometría es la geometría delimitadora formada por las coordenadas (X,Y) mínimas y máximas. Excepto en los casos especiales siguientes, los MBR de las geometrías forman un rectángulo delimitador.

- El MBR de cualquier punto es el propio punto, pues sus coordenadas X máxima y mínima son iguales, y sus coordenadas Y máxima y mínima son iguales.
- El MBR de una cadena lineal horizontal o vertical es una cadena lineal representada por el perímetro (los puntos finales) de la cadena lineal original.

Dimensión

La dimensión de una geometría puede ser igual a –1, 0, 1 ó 2. Esta es la descripción de cada dimensión:

–1 Está vacía

0 Carece de longitud y su área es 0 (cero)

1 Tiene una longitud mayor que 0 (cero) y su área es 0 (cero)

2 Tiene un área mayor que 0 (cero)

Los subtipos punto y multipunto tienen una dimensión igual a cero. Los puntos representan elementos dimensionales que se pueden representar mediante un conjunto individual de coordenadas, mientras que los multipuntos denotan datos que se deben representar con un conjunto de puntos.

Los subtipos cadena lineal y multilínea tienen una dimensión igual a uno. Sirven para representar tramos de carretera, sistemas fluviales y otros elementos geográficos de carácter lineal.

Los subtipos polígono y multipolígono tienen una dimensión igual a 2. El tipo de datos de polígono y multipolígono puede representar elementos geográficos cuyo perímetro delimita un área definible, tales como bosques, parcelas de terreno y lagos.

Identificador de sistemas de referencia espacial

El identificador de sistemas de referencia espacial determina qué sistema de referencia espacial se utiliza para representar una geometría.

Todos los sistemas de referencia espacial reconocidos por la base de datos se pueden acceder a través de la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Capítulo 3. Cómo utilizar DB2 Spatial Extender

Cómo utilizar DB2 Spatial Extender

El soporte y la utilización de DB2® Spatial Extender implica dos actividades principales: configurar DB2 Spatial Extender y llevar a cabo proyectos que utilizan datos espaciales. Este tema presenta las interfaces que puede utilizar para realizar tareas espaciales.

Interfaces para DB2 Spatial Extender y funcionalidad asociada

Varias interfaces permiten al usuario configurar DB2 Spatial Extender y crear proyectos que utilizan datos espaciales. Estas interfaces son:

- El Centro de control de DB2, una interfaz gráfica de usuario que incluye ventanas, cuadernos y selecciones de menús que dan soporte a DB2 Spatial Extender.
- Un procesador de línea de mandatos (CLP) proporcionado por DB2 Spatial Extender. Se denomina *CLP de db2se.*
- Programas de aplicación que llaman a procedimientos almacenados de DB2 Spatial Extender.

Otras interfaces permiten al usuario generar información espacial. Entre ellas se incluyen:

- Consultas SQL que el usuario somete desde un CLP de DB2, desde una ventana de consulta del Centro de control de DB2, o desde un programa de aplicación.
- Herramientas de visualización que producen información espacial en forma gráfica. Un ejemplo es ArcExplorer for DB2, que el Environmental Systems Research Institute (ESRI) creó para IBM®. ArcExplorer for DB2 se puede bajar del sitio Web de DB2 Spatial Extender:

  http://www.ibm.com/software/data/spatial/

Tareas que el usuario realiza para configurar DB2 Spatial Extender y crear proyectos

Esta sección proporciona una visión general de las tareas que el usuario tiene que realizar para establecer DB2 Spatial Extender y trabajar en proyectos que utilicen datos espaciales. Se incluye un caso práctico que ilustra las tareas. Las tareas se dividen en dos categorías:

- Configuración de DB2 Spatial Extender
- Creación de proyectos que utilizan datos espaciales

Configuración de DB2 Spatial Extender:

Esta sección enumera las tareas que el usuario puede realizar para configurar DB2 Spatial Extender y utiliza un caso práctico para ilustrar la manera cómo una empresa ficticia puede afrontar cada tarea.

Para configurar DB2 Spatial Extender:

1. Planifique y lleve a cabo la preparación (decida qué proyectos crear, decida qué interfaz o interfaces utilizar, seleccione personal para administrar DB2 Spatial Extender y crear los proyectos, etcétera).

Caso práctico: El entorno de sistemas de información de la compañía de seguros de inmuebles Safe Harbor incluye un sistema DB2 Universal Database™ y un sistema de archivos separado para datos espaciales únicamente. Como extensión, los resultados de consulta pueden incluir combinaciones de datos de ambos sistemas. Por ejemplo, una tabla de DB2 almacena información sobre ingresos, y un archivo del sistema de archivos contiene las ubicaciones de las sucursales de la compañía. Por lo tanto, es posible averiguar qué oficinas aportan ingresos de cantidades especificadas y, a continuación, determinar dónde están ubicadas esas oficinas. Pero los datos procedentes de los dos sistemas no se pueden integrar (por ejemplo, los usuarios no pueden unir columnas de DB2 con registros del sistema de archivos, y los servicios de DB2, tales como la optimización de consultas, no están disponibles para el sistema de archivos). Para solucionar estos problemas, Safe Harbor adquiere DB2 Spatial Extender Versión 8 y establece un nuevo Departamento de desarrollo espacial (llamado departamento Espacial de forma abreviada).

La primera misión del departamento Espacial consiste en incluir DB2 Spatial Extender en el entorno de DB2 de Safe Harbor:

- 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.
- Puesto que el equipo de administración tiene una sólida formación en UNIX®, decide utilizar el CLP de db2se para administrar DB2 Spatial Extender.
- 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.

2. Instale DB2 Spatial Extender.

   Caso práctico: El equipo de administración espacial instala DB2 Spatial Extender en una máquina UNIX dentro de un entorno DB2.

3. Si se dispone de DB2 Spatial Extender Versión 7, migre los datos espaciales a DB2 Versión 8.

   Caso práctico: El release de la Versión 8 es el primero que Safe Harbor ha adquirido. No es necesaria ninguna migración.

4. Configure la base de datos para alojar los datos espaciales. Ajuste los parámetros de configuración para asegurar que la base de datos tenga memoria y espacio suficiente para funciones espaciales, archivos de anotaciones cronológicas y aplicaciones de DB2 Spatial Extender.

   Caso práctico: Un miembro del equipo de administración espacial ajusta las características de los archivos de anotaciones cronológicas de transacción, el tamaño de la pila de aplicaciones y el tamaño de la pila de control de aplicaciones de acuerdo con los valores adecuados para los requisitos de DB2 Spatial Extender.

5. Configure los recursos espaciales para la base de datos. Estos recursos incluyen un catálogo del sistema, tipos de datos espaciales, funciones espaciales, un geocodificador y otros objetos. La tarea de configurar estos recursos se denomina *habilitación de la base de datos para operaciones espaciales.*

   El geocodificador proporcionado por DB2 Spatial Extender convierte direcciones de Estados Unidos en datos espaciales. Se denomina

DB2SE_USA_GEOCODER. Su organización y otras pueden proporcionar geocodificadores que conviertan direcciones de otros países y otros tipos de datos en datos espaciales.

Caso práctico: El equipo de administración espacial prepara los recursos que serán necesarios para los proyectos que planean.

- Un miembro del equipo emite un mandato para obtener los recursos que habilitan la base de datos para operaciones espaciales. Estos recursos incluyen el catálogo de DB2 Spatial Extender, tipos de datos espaciales, funciones espaciales, etcétera.
- 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. Safe Harbor no tiene intención de adquirir tales geocodificadores al menos durante unos meses. Por lo tanto, las primeras ubicaciones en las que reunirá datos serán en Estados Unidos.

Creación de proyectos que utilizan datos espaciales:

Después de configurar DB2 Spatial Extender, está preparado para emprender proyectos que utilicen datos espaciales. Esta sección enumera las tareas implicadas en la creación de un proyecto y continúa el caso práctico en el que la empresa de seguros de inmuebles Safe Harbor busca integrar datos comerciales y datos espaciales.

Para crear un proyecto que utilice datos espaciales:

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

   Caso práctico: El departamento se prepara para desarrollar el proyecto; por ejemplo:

   - El equipo de dirección establece estos objetivos para el proyecto:
     - Determinar dónde establecer nuevas sucursales
     - Fijar primas de riesgo en función de la proximidad de los clientes a zonas de riesgo (zonas con tasas altas de accidentes de tráfico, zonas con un nivel alto de criminalidad, zonas con riesgo de inundación, fallas tectónicas, etcétera.)
   - Este proyecto particular tratará con clientes y oficinas de Estados Unidos. Por lo tanto, el equipo de administración espacial decide:
     - Utilizar un sistema de coordenadas para Estados Unidos que DB2 Spatial Extender suministra. Se denomina GCS_NORTH_AMERICAN_1983.
     - Utilizar el geocodificador DB2SE_USA_GEOCODER, 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. Cree un sistema de coordenadas si el usuario necesita hacerlo así.

   Caso práctico: Porque Safe Harbor ha decidido utilizar GCS_NORTH_AMERICAN_1983, la compañía puede omitir este paso.
3. Decida si un sistema de referencia espacial existente se ajusta a sus necesidades. Si ninguno lo hace, cree uno.

   Un *sistema de referencia espacial* es un conjunto de valores de parámetros que incluye:

- Coordenadas que definen la máxima extensión de espacio posible al que se hace referencia mediante un rango determinado de coordenadas. El usuario necesita determinar el máximo rango de coordenadas posible que se pueda determinar desde el sistema de coordenadas que esté utilizando, y seleccionar o crear un sistema de referencia espacial que refleje este rango.
- El nombre del sistema de coordenadas del cual se obtienen las coordenadas.
- Los números utilizados en operaciones matemáticas para convertir coordenadas recibidas como datos de entrada en valores que se puedan procesar con máxima eficacia. Las coordenadas se guardan en su formato convertido y se devuelven al usuario en su formato original.

Caso práctico: DB2 Spatial Extender proporciona un sistema de referencia espacial, NAD83_SRS_1, que está diseñado para su utilización con GCS_NORTH_AMERICAN_1983. El equipo de administración espacial decide utilizar NAD83_SRS_1.

4. Cree columnas espaciales, según sea necesario. Se ha de tener en cuenta que en muchos casos, si una herramienta de visualización va a leer los datos de una columna espacial, la columna debe ser la única columna espacial de la tabla o vista a la que pertenece. De forma alternativa, si la columna es una de varias columnas espaciales en una tabla, se podría incluir en una vista que no tenga otras columnas espaciales, y las herramientas de visualización podrían leer los datos de esta vista.

   Caso práctico: El equipo de administración espacial define columnas para que contengan datos espaciales.

   - El equipo añade una columna LOCATION a la tabla CUSTOMERS. La tabla ya contiene las direcciones de los clientes. DB2SE_USA_GEOCODER las convertirá en datos espaciales. A continuación, DB2 almacenará estos datos en la columna LOCATION.
   - El equipo crea una tabla OFFICE_LOCATIONS y una tabla OFFICE_SALES para que contengan los datos que ahora se almacenan en el sistema de archivos separado. 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 obtenidos por el geocodificador irán dentro de una columna LOCATION de la tabla OFFICE_LOCATIONS, y los datos que definen las zonas irán dentro de una columna SALES_AREA de la tabla OFFICE_SALES.

5. Defina columnas espaciales para el acceso mediante herramientas de visualización, según sea necesario. El usuario realiza esta acción registrando las columnas en el catálogo DB2 Spatial Extender. Cuando se registra una columna espacial, DB2 Spatial Extender impone una restricción de que todos los datos de la columna deben pertenecer al mismo sistema de referencia espacial. Esta restricción refuerza la integridad de los datos—un requisito de la mayoría de las herramientas de visualización.

   Caso práctico: El equipo de administración espacial espera utilizar herramientas de visualización para producir el contenido de las columnas LOCATION y de la columna SALES_AREA gráficamente en un mapa. Por lo tanto, el equipo registra las tres columnas.

6. Llene con datos columnas espaciales:

   Para un proyecto que necesite que se importen datos espaciales, importe los datos.

   Para un proyecto que necesite un geocodificador:

   - Establezca, por adelantado, la información de control necesaria cuando se invoca un geocodificador.

- Como opción, configure el geocodificador para que se ejecute automáticamente cada vez que se añada una nueva dirección a la base de datos o que se actualice una dirección existente.

Ejecute el geocodificador en modalidad de proceso por lotes, según sea necesario.

Para un proyecto que requiera que se creen datos espaciales mediante una función espacial, ejecute esta función.

Caso práctico: El equipo de administración espacial llena la columna LOCATION de la tabla CUSTOMER, la tabla OFFICE_LOCATIONS, la tabla OFFICE_SALES y una nueva tabla HAZARD_ZONES:

- El equipo utiliza DB2SE_USA_GEOCODER para geocodificar direcciones de la tabla CUSTOMER. Las coordenadas producidas por la geocodificación se insertan en la columna LOCATION de la tabla.
- El equipo utiliza un programa de utilidad para cargar datos de oficina desde el sistema de archivos en un archivo. A continuación, el equipo importa estos datos a la nueva tabla OFFICE_LOCATIONS.
- El equipo crea una tabla HAZARD_ ZONES, registra sus columnas espaciales e importa los datos a la misma. Los datos proceden de un archivo suministrado por un proveedor de mapas.

7. Facilite el acceso a columnas espaciales, según sea necesario. Esto implica la definición de índices que posibiliten a DB2 acceder a datos espaciales de forma rápida y la definición de vistas que permitan a los usuarios recuperar eficazmente datos interrelacionados. Si se desea que las herramientas de visualización accedan a las columnas espaciales de la vista, es necesario registrar estas columnas en DB2 Spatial Extender también.

   Caso práctico: El equipo de administración espacial crea índices para las columnas registradas. A continuación, el equipo crea una vista que une columnas procedentes de las tablas CUSTOMERS y HAZARD ZONES. A continuación, registra las columnas espaciales de la vista.

8. Genere información espacial e información comercial asociada. Analice la información. Esta tarea implica consultar columnas espaciales y columnas no espaciales asociadas. En tales consultas, puede incluir funciones de DB2 Spatial Extender que devuelven una amplia variedad de información; por ejemplo, coordenadas que definen una zona de seguridad propuesta alrededor de una zona de vertidos peligrosa, o la distancia mínima entre el vertedero y el edificio público más cercano.

   Caso práctico: El equipo de análisis espacial ejecuta consultas para obtener información que le 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 zonas de riesgo.

Tareas relacionadas:

- "Instalación y configuración de Spatial Extender" en la página 25
- "Habilitación de una base de datos para operaciones espaciales" en la página 54
- "Registro de un geocodificador" en la página 56
- "Configuración de una base de datos para alojar datos espaciales" en la página 49
- "Importación de datos de formas a una tabla nueva o existente" en la página 90
- "Importación de datos de transferencia SDE a una tabla nueva o existente" en la página 92
- "Configuración de las operaciones de geocodificación" en la página 97

- "Configuración de un geocodificador para que se ejecute automáticamente" en la página 99
- "Ejecución de un geocodificador en modalidad de proceso por lotes" en la página 100
- "Exportación de datos a un archivo de transferencia SDE" en la página 94
- "Selección o creación de sistemas de coordenadas" en la página 67
- "Registro de columnas espaciales" en la página 87
- "Creación de columnas espaciales" en la página 86
- "Creación de índices reticulares espaciales" en la página 111
- "Cómo llamar a los procedimientos almacenados de DB2 Spatial Extender desde una aplicación" en la página 142
- "Cómo incluir el archivo de cabecera de DB2 Spatial Extender en aplicaciones espaciales" en la página 141

Información relacionada:

- "Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos" en la página 131

Parte 2. Configuración de DB2 Spatial Extender

Capítulo 4. Iniciación a DB2 Spatial Extender

Este capítulo proporciona instrucciones para instalar y configurar Spatial Extender para AIX, HP-UX, Windows NT, Windows 2000, Linux, Linux para z/OS y Entornos operativos Solaris. Este capítulo también explica cómo resolver algunos problemas de instalación y configuración con los que se podría encontrarse al invocar Spatial Extender.

Instalación y configuración de Spatial Extender—Pasos

Esta sección contiene detalles sobre los temas siguientes:

- Requisitos del sistema para instalar Spatial Extender
- Instrucciones de instalación para plataformas UNIX y Windows
- Explicación sobre cómo crear un entorno de instancias de DB2 Spatial Extender
- Verificación de la instalación de Spatial Extender
- Indicaciones para la resolución de problemas del programa de ejemplo de instalación

Instalación y configuración de Spatial Extender

Un sistema DB2 Spatial Extender consta de DB2 Universal Database, DB2 Spatial Extender, y, en la mayoría de casos, un geonavegador. El geonavegador no es obligatorio, pero es útil para representar visualmente los resultados de las consultas espaciales, generalmente en forma de mapas. Las bases de datos habilitadas para operaciones espaciales se ubican en el servidor. Puede utilizar las aplicaciones cliente para acceder a los datos espaciales mediante los procedimientos almacenados y las consultas espaciales de DB2 Spatial Extender. También puede configurar DB2 Spatial Extender en un entorno autónomo, que es una configuración en la que el cliente y el servidor residen en la misma máquina. Tanto en configuraciones cliente-servidor como en configuraciones autónomas, puede ver los datos espaciales con un geonavegador, tal como las suites de herramientas ArcExplorer for DB2 o ArcGIS de ESRI que se ejecutan con ArcSDE.

Puede bajar una copia gratuita de ArcExplorer for DB2 desde el sitio Web de DB2 Spatial Extender de IBM:

http://www.ibm.com/software/data/spatial/

Requisitos previos:

Antes de instalar DB2 Spatial Extender, debe tener instalado y configurado el software de DB2 en el cliente y en el servidor.

Procedimiento:

1. Asegúrese de que el sistema cumpla todos los requisitos de software.
2. Instale Spatial Extender. Los pasos varían en función del sistema operativo:
   - Windows
   - AIX
   - HP-UX
   - Entorno operativo Solaris

- Linux

3. Para plataformas UNIX: Cree un entorno de instancias de DB2 Spatial Extender.
4. Verifique la instalación.
5. Si es necesario, consulte las indicaciones para la resolución de problemas y realice las acciones apropiadas para corregir cualquier problema.
6. Si desea acceder a la documentación de DB2 desde el sistema y todavía no ha instalado el Centro de información de DB2, consulte uno de los siguientes manuales: Instalación del Centro de información de DB2 (UNIX)" o Instalación del Centro de información de DB2 (Windows). El Centro de información de DB2 contiene documentación para DB2 Universal Database y productos relacionados con DB2.

Conceptos relacionados:

- "Requisitos del sistema para la instalación de Spatial Extender" en la página 26

Tareas relacionadas:

- "Instalación de DB2 Spatial Extender para Windows" en la página 27
- "Instalación de DB2 Spatial Extender para AIX" en la página 29
- "Instalación de DB2 Spatial Extender para HP-UX" en la página 31
- "Instalación de DB2 Spatial Extender para el Entorno operativo Solaris" en la página 33
- "Instalación de DB2 Spatial Extender para Linux" en la página 35
- "Creación del entorno de instancias de DB2 Spatial Extender" en la página 37
- "Verificación de la instalación de Spatial Extender" en la página 39
- "Resolución de problemas de instalación" en la página 41
- "Instalación del Centro de información de DB2 mediante el Asistente de instalación de DB2 (UNIX)" en el manual *Infrastructure Topics (DB2 Common Files)*
- "Instalación del Centro de información de DB2 mediante el Asistente de instalación de DB2 (Windows)" en el manual *Infrastructure Topics (DB2 Common Files)*
- "Bajada de ArcExplorer for DB2" en la página 41

Información relacionada:

- "CD para datos y mapas de DB2 Spatial Extender" en la página 43

Requisitos del sistema para la instalación de Spatial Extender

Antes de instalar DB2® Spatial Extender, asegúrese de que el sistema cumpla todos los requisitos de software y de espacio descritos más abajo.

Sistemas operativos:

Puede instalar DB2 Spatial Extender en las versiones de 32 y 64 bits de Windows®, AIX®, HP-UX, el Entorno Operativo Solaris y Linux para Intel. Spatial Extender tiene soporte para Linux para S/390® (32 bits) pero no tiene soporte para Linux para zSeries® (64 bits).

Requisitos de software:

Para instalar Spatial Extender, debe tener instalado y configurado en el servidor el siguiente software de DB2:

Software del servidor

DB2 Universal Database™ Enterprise Server Edition Versión 8.2 debe estar instalado en el sistema *antes* de instalar DB2 Spatial Extender. Si piensa utilizar el Centro de control de DB2, cree y configure el Servidor de administración de DB2 (DAS). Para obtener más información sobre la creación y la configuración del DAS, consulte el manual *IBM® DB2 Universal Database Administration Guide: Implementation*.

Software del cliente espacial

Si instala DB2 Spatial Extender en Windows, la instalación por omisión de Spatial Extender incluye el cliente espacial. Para DB2 Spatial Extender en AIX, HP-UX, Entorno Operativo Solaris, Linux para Intel o Linux para S/390, puede instalar el cliente espacial cuando instale el servidor DB2 con el cliente de administración o de desarrollo de aplicaciones. Si no instala estos clientes, debe instalar el cliente espacial manualmente seleccionando la opción Instalación personalizada.

Requisitos de espacio en disco:

Para instalar Spatial Extender, el sistema debe cumplir los requisitos de espacio en disco que se listan en la tabla siguiente. El código de bibliotecas de DB2 Spatial Extender integra el código de DB2 Geodetic Extender pero no integra la clave de licencia de Geodetic.

Tabla 1. Requisitos de espacio en disco para DB2 Spatial Extender

Software de DB2 Spatial Extender	Espacio en disco
Software de servidor para DB2 Spatial Extender:	Espacio en disco total de 596 MB:
• Código de bibliotecas de servidor Spatial Extender y Geodetic Extender, datos de referencia de ejemplo del geocodificador y documentación	• 33 MB
• Opcionalmente disponible en un CD independiente: datos de referencia del geocodificador (Estados Unidos)	• 563 MB

La Tabla 1 especifica el espacio en disco necesario para instalar DB2 Universal Database y DB2 Spatial Extender en una instalación típica para Windows o con componentes preseleccionados en AIX, HP-UX, Entorno Operativo Solaris, Linux para Intel y Linux para S/390. Si está instalando DB2 Spatial Extender o ha instalado DB2 Universal Database con un tipo de instalación diferente, los cálculos de espacio en disco variarán.

Cuando el sistema cumpla todos los requisitos de software y de espacio en disco, puede instalar Spatial Extender.

Instalación de DB2 Spatial Extender para Windows

Esta tarea forma parte de la tarea más amplia de Configuración de DB2 Spatial Extender.

Puede instalar DB2 Spatial Extender en sistemas operativos Windows mediante la utilización del Asistente de instalación de DB2 o un archivo de respuestas.

Recomendación: utilice el Asistente de instalación de DB2 para instalar Spatial Extender. El asistente de instalación proporciona una interfaz gráfica de fácil utilización con ayuda de instalación, creación automatizada de grupos y usuarios, configuración de protocolo y creación de instancias.

Si está utilizando el Asistente de instalación de DB2 para instalar Spatial Extender, puede pulsar **Cancelar** en cualquier momento durante la instalación para salir del proceso.

Requisitos previos:

Antes de instalar el producto DB2 Spatial Extender, debe tener instalado un producto de servidor de DB2 Versión 8.

Procedimiento:

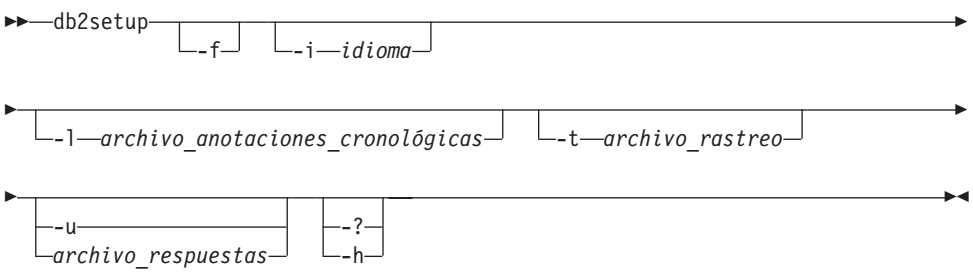
Para instalar Spatial Extender para Windows mediante el Asistente de instalación de DB2:

1. Inserte el CD de Spatial Extender en la unidad de CD. Se abre el Área de ejecución de instalación de DB2, una interfaz desde la que puede instalar DB2 Spatial Extender.
2. Pulse **Instalar productos**.
3. Seleccione DB2 Spatial Extender como el producto que desea instalar y pulse **SIGUIENTE**. Se inicia el Asistente de instalación de DB2. Pulse **SIGUIENTE**. Utilice el Asistente de instalación de DB2 que le guiará durante el proceso de configuración y durante los demás pasos de la instalación. En cualquier momento durante la instalación puede pulsar **Ayuda** para iniciar la ayuda en línea de la instalación.

Para instalar Spatial Extender para Windows mediante el archivo de respuestas:

1. Inicie una sesión en el sistema con la cuenta de usuario que desea utilizar para realizar la instalación.
2. Inserte el CD de Spatial Extender. Consulte el manual *Suplemento de instalación y configuración de DB2* para obtener más información.
3. Ejecute el programa de instalación emitiendo el siguiente mandato desde la línea de mandatos:

Mandato db2setup



Donde:

-f Hace que se detengan todos los procesos de DB2 antes de comenzar la instalación.

-i (*idioma***)**
Código de dos letras que indica el idioma utilizado durante el proceso de instalación.

-l (*archivo_anotaciones_cronológicas* **)**
Vía de acceso completa y nombre del archivo de anotaciones cronológicas que se debe utilizar.

-t (*archivo_rastreo***)**
Genera un archivo, calificado al completo, que contiene información de rastreo de la instalación.

-u (*archivo_respuestas***)**
Especifica el nombre del archivo de respuestas, calificado al completo. Si ha modificado o renombrado el archivo de respuestas de ejemplo que se proporciona, asegúrese de que este parámetro coincida con el nuevo nombre. Este parámetro es obligatorio. El archivo de respuestas está situado en el directorio db2\Windows\samples\db2gse.rsp del CD de instalación de DB2 Spatial Extender.

-?, -h Produce información sobre el modo de utilizar el mandato.

4. Una vez finalizada la instalación, examine los mensajes del archivo de anotaciones cronológicas.

Conceptos relacionados:

- "Requisitos del sistema para la instalación de Spatial Extender" en la página 26

Tareas relacionadas:

- "Creación y edición de un archivo de respuestas (Windows)" en el manual *Suplemento de instalación y configuración*
- "Verificación de la instalación de Spatial Extender" en la página 39
- "Resolución de problemas de instalación" en la página 41

Instalación de DB2 Spatial Extender para AIX

Puede instalar DB2 Spatial Extender para AIX utilizando el Asistente de instalación de DB2, el script db2_install o la herramienta SMIT (System Management Interface Tool).

Recomendación: utilice el Asistente de instalación de DB2 para instalar Spatial Extender. El Asistente de instalación proporciona una interfaz gráfica de fácil utilización con ayuda de instalación, creación automatizada de grupos y usuarios, configuración de protocolo y creación de instancias. Si decide no utilizar el asistente, puede instalar Spatial Extender mediante el script db2_install o la herramienta SMIT (System Management Interface Tool) de AIX. La utilización de SMIT para instalar Spatial Extender sólo es recomendable para usuarios experimentados en los casos en que sea necesario un mayor control manual sobre el proceso de instalación.

Requisitos previos:

Antes de instalar Spatial Extender en AIX:

- Asegúrese de que el sistema cumpla todos los requisitos de espacio en disco, memoria y software.
- Actualice los parámetros de configuración y reinicie el sistema para todos los clientes y servidores DB2 en AIX.

- Debe tener instalado el producto de servidor de DB2 Versión 8 si está instalando en un servidor o en un entorno autónomo.

  Nota: Compruebe si DB2 Spatial Client ya está instalado. El cliente de Spatial Extender y los componentes de ejemplo ya están disponibles con el cliente y el servidor DB2.Puede instalar estos componentes espaciales utilizando el tipo de instalación personalizada de DB2, seleccionando la función **Cliente de Spatial Extender** bajo **Soporte al cliente** y la función **Ejemplos de Spatial Extender** bajo **Herramientas de desarrollo de aplicaciones**. Si sólo necesita las funciones del cliente espacial y ya ha instalado estos componentes espaciales con DB2, no es necesario que realice el procedimiento de instalación de DB2 Spatial Extender siguiente:
- Debe tener autorización root.

Procedimiento:

Para instalar Spatial Extender utilizando el Asistente de instalación de DB2:

1. Inicie la sesión como usuario con autorización root.
2. Inserte y monte el CD de Spatial Extender. Se abre el Área de ejecución de instalación de DB2, una interfaz desde la que puede instalar DB2 Spatial Extender. Para obtener información acerca de cómo montar un CD, consulte el manual *Suplemento de instalación y configuración de DB2*.
3. Seleccione DB2 Spatial Extender como el producto que desea instalar y pulse **SIGUIENTE**.
4. Se abrirá la ventana Asistente de instalación de DB2. Utilice el Asistente de instalación de DB2 que le guiará durante el proceso de configuración y durante los demás pasos de la instalación. En cualquier momento durante la instalación puede pulsar **Ayuda** para iniciar la ayuda en línea de la instalación.

Para instalar DB2 Spatial Extender utilizando el script db2_install:

1. Inicie la sesión como usuario con autorización root.
2. Inserte y monte el CD apropiado.
3. Ejecute el mandato **./db2_install** para iniciar el script db2_install. El script db2_install reside en el directorio raíz del CD del producto DB2 Versión 8. El script db2_install le solicitará la palabra clave del producto.
4. Escriba **DB2.GSE** para instalar DB2 Spatial Extender.

Para instalar DB2 Spatial Extender utilizando la herramienta SMIT (System Management Interface Tool):

1. Inicie la sesión como usuario con autorización root.
2. Inserte y monte el CD de Spatial Extender.
3. Ejecute el mandato **smit install_latest**.
4. Escriba `/cdrom/db2` en **INPUT device/directory** para el campo de software.
5. Pulse **EJECUTAR** o pulse Intro para verificar que existe el directorio de instalación.
6. En el campo **Software a instalar**, especifique si deben instalarse los componentes de cliente o de servidor.

   Nota: Consulte el archivo ComponentList.htm en el CD de DB2 Spatial Extender para obtener una lista completa de los componentes que es necesario instalar para DB2 Spatial Extender.

7. Pulse **Ejecutar** o pulse Intro. Se le solicitará que confirme los parámetros de instalación.
8. Pulse Intro para confirmar.
9. Finalice la sesión.

Cuando finalice la instalación, Spatial Extender estará instalado en el directorio /usr/opt/db2_08_01 junto con los otros productos DB2.

Después de instalar Spatial Extender, cree el entorno de instancias de DB2 si no lo ha hecho aún y, a continuación, verifique la instalación.

Conceptos relacionados:

- "Requisitos del sistema para la instalación de Spatial Extender" en la página 26

Tareas relacionadas:

- "Instalación de un producto de DB2 mediante la SMIT (AIX)" en el manual *Suplemento de instalación y configuración*
- "Montaje del CD-ROM (AIX)" en el manual *Guía rápida de iniciación para servidores DB2*
- "Instalación de un producto de DB2 mediante el script db2_install (UNIX)" en el manual *Suplemento de instalación y configuración*
- "Verificación de la instalación de Spatial Extender" en la página 39
- "Resolución de problemas de instalación" en la página 41

Instalación de DB2 Spatial Extender para HP-UX

Puede instalar Spatial Extender utilizando el Asistente de instalación de DB2®, el script db2_install o el mandato **swinstall**.

Recomendación: utilice el Asistente de instalación de DB2 para instalar Spatial Extender. El asistente de instalación proporciona una interfaz gráfica de fácil utilización con ayuda de instalación, creación automatizada de grupos y usuarios, configuración de protocolo y creación de instancias. Si decide no utilizar el asistente, puede instalar Spatial Extender para HP-UX utilizando el script db2_install o el mandato **swinstall**. La utilización del mandato **swinstall** de HP-UX para instalar Spatial Extender sólo es recomendable para usuarios experimentados en los casos en que sea necesario un mayor control manual sobre el proceso de instalación.

Requisitos previos:

Antes de instalar el producto DB2 Spatial Extender para HP-UX:

- Asegúrese de que el sistema cumpla todos los requisitos de memoria, software y hardware.
- Debe tener instalado el producto de servidor de DB2 Versión 8.

Nota: Compruebe si DB2 Spatial Client ya está instalado. El cliente de Spatial Extender y los componentes de ejemplo ya están disponibles con el cliente y el servidor DB2.Puede instalar estos componentes espaciales utilizando el tipo de instalación personalizada de DB2, seleccionando la función **Cliente de Spatial Extender** bajo **Soporte al cliente** y la función **Ejemplos de Spatial Extender** bajo **Herramientas de desarrollo de aplicaciones**. Si sólo necesita las funciones del cliente espacial y ya ha instalado estos

componentes espaciales con DB2, no es necesario que realice el procedimiento de instalación de DB2 Spatial Extender siguiente:

- Actualice los parámetros de configuración y reinicie el sistema para todos los clientes y servidores DB2 en HP-UX
- Debe tener autorización root.

Procedimiento:

Para instalar Spatial Extender para HP-UX mediante el Asistente de instalación de DB2:

1. Inserte y monte el CD de DB2 Spatial Extender. Se abre el Área de ejecución de instalación de DB2, una interfaz desde la que puede instalar DB2 Spatial Extender.
2. Seleccione DB2 Spatial Extender como el producto que desea instalar y pulse **SIGUIENTE**. Se inicia el Asistente de instalación de DB2. Pulse **SIGUIENTE**. Utilice el Asistente de instalación de DB2 que le guiará durante el proceso de configuración y durante los demás pasos de la instalación. En cualquier momento durante la instalación puede pulsar **Ayuda** para iniciar la ayuda en línea de la instalación.

Para instalar Spatial Extender para HP-UX utilizando el script db2_install:

1. Inicie la sesión como usuario con autorización root.
2. Inserte y monte el CD apropiado.
3. Ejecute el mandato **./db2_install** para iniciar el script db2_install. El script db2_install reside en el directorio raíz del CD del producto DB2 Versión 8. El script db2_install le solicitará la palabra clave del producto.
4. Escriba **DB2.GSE** para instalar DB2 Spatial Extender.

Para instalar Spatial Extender para HP-UX utilizando el mandato **swinstall**:

1. Inicie la sesión como usuario con autorización root.
2. Inserte y monte el CD de Spatial Extender.
3. Ejecute el programa swinstall utilizando el siguiente mandato:

   ```
   swinstall -x autoselect_dependencies=true
   ```

   Este mandato abre la ventana Selección de software y la ventana Especificar fuente. Si es necesario, cambie el valor del campo **Nombre de sistema principal fuente** en la ventana Especificar fuente.
4. En el campo **Vía de acceso de depósito fuente**, escriba */cdrom*/db2/hpux, donde */cdrom* representa el directorio de montaje del CD.
5. Pulse **Bien** para volver a la ventana Selección de software. La ventana Selección de software contiene una lista de software disponible para instalar.
6. Seleccione los productos de los que tenga licencia de instalación.
7. Seleccione **Seleccionar para instalar** desde el menú **Acciones** para seleccionar el producto que desea instalar. Aparece el mensaje:

   ```
   Además del software que acaba de seleccionar, se ha seleccionado otro software
   automáticamente para resolver dependencias.  Este mensaje
   no aparecerá otra vez.
   ```
8. Seleccione **Bien**.
9. Seleccione **Instalar (análisis)** en el menú **Acciones** para empezar a instalar el producto y para abrir la ventana Instalar análisis.

10. Seleccione **Bien** en la ventana Instalar análisis cuando el campo **Estado** muestre un mensaje de Listo.
11. Seleccione **Sí** en la ventana Confirmación para confirmar que desea instalar el software.

    Vea la ventana Instalar para leer los datos de proceso mientras se está instalando el software, hasta que el campo **Estado** indique Listo y se abra la ventana Nota. El programa swinstall carga el archivo y ejecuta los scripts de control para el archivo.
12. Seleccione **Salir** en el menú **Archivo** para salir de swinstall.

Cuando se complete la instalación, Spatial Extender estará instalado en el directorio /opt/IBM/db2/V8.1 junto con los otros productos DB2.

Después de instalar Spatial Extender, cree el entorno de instancias de DB2 si no lo ha hecho aún y, a continuación, verifique la instalación.

Conceptos relacionados:

- "Requisitos del sistema para la instalación de Spatial Extender" en la página 26

Tareas relacionadas:

- "Instalación de un producto de DB2 mediante swinstall (HP-UX)" en el manual *Suplemento de instalación y configuración*
- "Cómo montar el CD-ROM (HP-UX)" en el manual *Guía rápida de iniciación para servidores DB2*
- "Instalación de un producto de DB2 mediante el script db2_install (UNIX)" en el manual *Suplemento de instalación y configuración*
- "Verificación de la instalación de Spatial Extender" en la página 39
- "Resolución de problemas de instalación" en la página 41

Instalación de DB2 Spatial Extender para el Entorno operativo Solaris

Puede instalar Spatial Extender utilizando el Asistente de instalación de DB2®, el script db2_install o el mandato **pkgadd**.

Recomendación: utilice el Asistente de instalación de DB2 para instalar DB2 Spatial Extender. El asistente de instalación proporciona una interfaz gráfica de fácil utilización con ayuda de instalación, creación automatizada de grupos y usuarios, configuración de protocolo y creación de instancias. Si decide no utilizar el asistente, puede instalar Spatial Extender para utilizando el script db2_install o el mandato **pkgadd** del Entorno Operativo Solaris. La utilización del mandato **pkgadd** del Entorno Operativo Solaris sólo es recomendable para usuarios experimentados en los casos en que sea necesario un mayor control manual sobre el proceso de instalación.

DB2 Spatial Extender consta de diferentes funciones y componentes, denominados paquetes en el Entorno Operativo Solaris. Cuando instala Spatial Extender utilizando el mandato **pkgadd**, debe instalar cada paquete necesario y cada paquete asociado para las funciones opcionales que desee utilizar. El archivo ComponentList.htm, del CD de DB2 Spatial Extender, contiene una lista completa de los componentes que es necesario instalar para DB2 Spatial Extender. El archivo ComponentList.htm file se encuentra en */cdrom*/db2/solaris, donde */cdrom* es el punto de montaje del CD de DB2 Spatial Extender.

Requisitos previos:

Antes de instalar el producto DB2 Spatial Extender para los Entornos operativos Solaris:

- Asegúrese de que el sistema cumpla todos los requisitos de memoria, software y hardware.
- Debe tener instalado el producto de servidor de DB2 Versión 8 si está instalando en un servidor o en un entorno autónomo.

  Nota: Compruebe si DB2 Spatial Client ya está instalado. El cliente de Spatial Extender y los componentes de ejemplo ya están disponibles con el cliente y el servidor DB2.Puede instalar estos componentes espaciales utilizando el tipo de instalación personalizada de DB2, seleccionando la función **Cliente de Spatial Extender** bajo **Soporte al cliente** y la función **Ejemplos de Spatial Extender** bajo **Herramientas de desarrollo de aplicaciones**. Si sólo necesita las funciones del cliente espacial y ya ha instalado estos componentes espaciales con DB2, no es necesario que realice el procedimiento de instalación de DB2 Spatial Extender siguiente:
- Actualice los parámetros de configuración y reinicie el sistema para todos los clientes y servidores DB2 en el Entorno Operativo Solaris.
- Debe tener autorización root.

Procedimiento:

Para instalar DB2 Spatial Extender para Entornos Operativos Solaris utilizando el Asistente de instalación de DB2:

1. Inicie la sesión como usuario con autorización root.
2. Inserte y monte el CD de DB2 Spatial Extender. Se abre el Área de ejecución de instalación de DB2, una interfaz desde la que puede instalar DB2 Spatial Extender. Para obtener información sobre cómo montar un CD, consulte el manual *DB2 para UNIX Guía rápida de iniciación*.
3. Seleccione **Spatial Extender** como el producto que desea instalar y pulse **SIGUIENTE**.
4. Se inicia el Asistente de instalación de DB2. Utilice el Asistente de instalación de DB2 que le guiará durante el proceso de configuración y durante los demás pasos de la instalación. En cualquier momento durante la instalación, puede pulsar **AYUDA** para iniciar la ayuda en línea de la instalación.

Para instalar DB2 Spatial Extender utilizando el script db2_install:

1. Inicie la sesión como usuario con autorización root.
2. Inserte y monte el CD apropiado.
3. Ejecute el mandato **./db2_install** para iniciar el script db2_install. El script db2_install reside en el directorio raíz del CD del producto DB2 Versión 8. El script db2_install le solicitará la palabra clave del producto.
4. Escriba **DB2.GSE** para instalar DB2 Spatial Extender.

Para instalar DB2 Spatial Extender para Solaris utilizando el mandato **pkgadd**:

1. Inicie la sesión como usuario con autorización root.
2. Inserte y monte el CD de DB2 Spatial Extender.

3. Identifique los paquetes necesarios y los paquetes opcionales que desea instalar. Consulte el archivo ComponentList.htm, del CD de Spatial Extender, para obtener una lista completa de los componentes que es necesario instalar para DB2 Spatial Extender.
4. Ejecute el mandato **pkgadd** para cada paquete que desee instalar escribiendo:
   ```
   pkgadd nombre_paquete
   ```
   En este mandato, *nombre_paquete* es el paquete que desea instalar.
   Por ejemplo, si desea instalar el Soporte base de servidor de Spatial Extender, necesita instalar el paquete db2gssg81 entrando el siguiente mandato:
   ```
   pkgadd db2gssg81
   ```

Cuando se complete la instalación, el software de Spatial Extender estará instalado en el directorio /opt/IBM/db2/V8.1.

Cree el entorno de instancias de DB2 si no lo ha hecho aún y, a continuación, verifique la instalación.

Conceptos relacionados:

- "Requisitos del sistema para la instalación de Spatial Extender" en la página 26

Tareas relacionadas:

- "Instalación de un producto de DB2 mediante pkgadd (Entorno operativo Solaris)" en el manual *Suplemento de instalación y configuración*
- "Instalación de un producto de DB2 mediante el script db2_install (UNIX)" en el manual *Suplemento de instalación y configuración*
- "Montaje del CD-ROM (Entorno operativo Solaris)" en el manual *Guía rápida de iniciación para servidores DB2*
- "Verificación de la instalación de Spatial Extender" en la página 39
- "Resolución de problemas de instalación" en la página 41

Instalación de DB2 Spatial Extender para Linux

Puede instalar DB2 Spatial Extender para Linux utilizando el Asistente de instalación de DB2, el script db2_install o el mandato **rpm**.

Recomendación: utilice el Asistente de instalación de DB2 para instalar Spatial Extender. El asistente de instalación proporciona una interfaz gráfica de fácil utilización con ayuda de instalación, creación automatizada de grupos y usuarios, configuración de protocolo y creación de instancias. Si decide no utilizar el asistente, puede instalar Spatial Extender utilizando el script db2_install o el mandato **rpm**. La utilización del mandato **rpm** de Linux para instalar Spatial Extender sólo es recomendable para usuarios experimentados cuando sea necesario un mayor control manual sobre el proceso de instalación.

Requisitos previos:

Antes de instalar el producto DB2 Spatial Extender para Linux:

- Asegúrese de que el sistema cumpla todos los requisitos de memoria, software y hardware.
- Asegúrese de que tenga instalado el producto del servidor DB2 Versión 8 si está realizando la instalación en un entorno de servidor o en un entorno autónomo.

Nota:

- Compruebe si DB2 Spatial Client ya está instalado. DB2 Spatial Extender Client y los componentes de ejemplo están disponibles con el cliente y el servidor DB2. Puede instalar estos componentes espaciales utilizando el tipo de instalación personalizada de DB2 y seleccionando la función **Spatial Extender Client** bajo **Soporte de clientes** y seleccionando la función **Ejemplos de Spatial Extender** bajo **Herramientas de desarrollo de aplicaciones**. Si sólo necesita las funciones del cliente espacial y ya ha instalado estos componentes espaciales con DB2, no es necesario que realice el procedimiento de instalación de DB2 Spatial Extender siguiente:
- Actualice los parámetros de configuración y reinicie el sistema para todos los clientes y servidores DB2 de Linux.
- Asegúrese de tener autorización root.

Procedimiento:

Para instalar DB2 Spatial Extender utilizando el Asistente de instalación de DB2:

1. Inicie la sesión como usuario con autorización root.
2. Inserte y monte el CD de DB2 Spatial Extender. Se abre el Área de ejecución de instalación de DB2, una interfaz desde la que puede instalar DB2 Spatial Extender. Para obtener información acerca de cómo montar un CD, consulte el manual *Suplemento de instalación y configuración de DB2*.
3. Pulse **Instalar productos**.
4. Seleccione **Spatial Extender** como el producto que desea instalar y pulse **SIGUIENTE**.
5. Seleccione la opción que desee en la ventana Asistente de instalación de DB2. Tiene la opción de instalar **DB2 Spatial Extender** o **DB2 Application Development Client**.
   - Si sólo necesita las funciones del cliente espacial, seleccione **DB2 Application Development Client** y seleccione las funciones siguientes:
     - La función **Spatial Extender Client** bajo **Soporte de clientes**
     - Opcional: La función **Ejemplos de Spatial Extender** bajo **Herramientas de desarrollo de aplicaciones**
   - Si necesita las funciones tanto del servidor como del cliente espacial, seleccione **DB2 Spatial Extender** y seleccione las funciones siguientes:
     - La función **Soporte del servidor base de Spatial Extender** bajo **Soporte de servidores**
     - La función **Spatial Extender Client** bajo **Soporte de Spatial Extender Client**
     - Opcional: La función **Ejemplos de Spatial Extender** bajo **Herramientas de desarrollo de aplicaciones**

   Utilice el Asistente de instalación de DB2 que le guiará durante el proceso de configuración y durante los demás pasos de la instalación. En cualquier momento durante la instalación puede pulsar **Ayuda** para iniciar la ayuda en línea de la instalación.

Para instalar DB2 Spatial Extender utilizando el script db2_install:

1. Inicie la sesión como usuario con autorización root.
2. Inserte y monte el CD apropiado.

3. Ejecute el mandato **./db2_install** para iniciar el script db2_install. El script db2_install reside en el directorio raíz del producto DB2 Versión 8. El script db2_install le solicitará la palabra clave del producto.
4. Escriba **DB2.GSE** para instalar DB2 Spatial Extender.

Para instalar Spatial Extender para Linux utilizando el mandato **rpm**:

1. Inicie la sesión como usuario con autorización root.
2. Habilite el sistema para la instalación de DB2 para Linux. Consulte el manual *Suplemento de instalación y configuración de DB2* para obtener más información.
3. Inserte y monte el CD de DB2 Spatial Extender.
4. Identifique los paquetes necesarios y los paquetes opcionales que desea instalar.

   Nota: Consulte el archivo ComponentList.htm, del CD de DB2 Spatial Extender, para obtener una lista completa de los componentes que es necesario instalar para DB2 Spatial Extender.
5. Ejecute el mandato **rpm** para cada paquete que desea instalar:

   ```
   rpm -ivh nombre_paquete
   ```

   Por ejemplo, si desea instalar el servidor, instale el paquete IBM_db2gssg81-8.1.0-0.i386.rpm entrando el mandato siguiente:

   ```
   rpm -IBM_db2gssg81-8.1.0-0.i386.rpm
   ```

Cuando se complete la instalación, Spatial Extender estará instalado en el directorio /opt/IBM/db2/V8.1 junto con los otros productos DB2.

Después de instalar Spatial Extender, cree el entorno de instancias de DB2 si no lo ha hecho aún y, a continuación, verifique la instalación.

Conceptos relacionados:

- "Requisitos del sistema para la instalación de Spatial Extender" en la página 26

Tareas relacionadas:

- "Instalación de un producto de DB2 mediante rpm (Linux)" en el manual *Suplemento de instalación y configuración*
- "Montaje del CD-ROM (Linux)" en el manual *Guía rápida de iniciación para servidores DB2*
- "Instalación de un producto de DB2 mediante el script db2_install (UNIX)" en el manual *Suplemento de instalación y configuración*
- "Verificación de la instalación de Spatial Extender" en la página 39
- "Resolución de problemas de instalación" en la página 41

Creación del entorno de instancias de DB2 Spatial Extender

Esta tarea forma parte de la tarea más amplia de instalar Spatial Extender.

Esta sección sólo es aplicable para plataformas UNIX.

DB2 Spatial Extender se puede utilizar con cualquier instancia de DB2 que se cree después de instalar el código de Spatial Extender.

El mandato **db2icrt** se utiliza para crear nuevas instancias de DB2. Todas las nuevas instancias de DB2 que crea después de instalar DB2 Spatial Extender incluyen DB2 Spatial Extender en el entorno de instancia.

Las instancias de DB2 creadas antes de instalar Spatial Extender no incluyen DB2 Spatial Extender en sus entornos de instancia. Para actualizar instancias de DB2 existentes, utilice el mandato **db2iupdt**. Si está utilizando el Centro de control de DB2 y ha creado una instancia para el Servidor de administración de DB2 antes de instalar DB2 Spatial Extender, debe actualizar esta instancia.

Procedimiento:

Para actualizar una instancia utilizando el mandato **db2iupdt**:

1. Inicie la sesión como usuario con autorización root.
2. Ejecute el siguiente mandato:

   ```
   DB2DIR/instance/db2iupdt -a AuthType -u FencedID InstName
   ```

   Donde:

   DB2DIR
   Es el directorio de instalación de DB2.
   - En AIX, el directorio de instalación de DB2 es /usr/opt/db2_08_01
   - En todos los demás sistemas operativos basados en UNIX, el directorio de instalación es /opt/IBM/db2/V8.1

   -a AuthType
   Representa el tipo de autenticación para la instancia. AuthType puede ser uno de los siguientes: SERVER, CLIENT, DCS, SERVER_ENCRYPT, DCS_ENCRYPT. SERVER es el valor por omisión. Este parámetro es opcional.

   -u FencedID
   Representa el nombre del usuario bajo el que las funciones definidas por el usuario delimitado (las UDF) y los procedimientos almacenados delimitados se ejecutarán. Este distintivo no es necesario si está creando una instancia en un cliente DB2. Especifique el nombre del usuario delimitado que ha creado.

   InstName
   Representa el nombre de instancia. El nombre de la instancia debe ser el mismo que el nombre del usuario que es propietario de la instancia. Especifique el nombre del usuario que es propietario de la instancia que ha creado. La instancia se creará en el directorio inicial del usuario que es propietario de la instancia.

Para crear una instancia utilizando **db2icrt**:

1. Inicie la sesión como usuario con autoridad root.
2. Ejecute el siguiente mandato:

   ```
   DB2DIR/instance/db2icrt -a AuthType -u FencedID InstName
   ```

   Donde:

   DB2DIR
   Es el directorio de instalación de DB2.
   - En AIX, el directorio de instalación de DB2 es /usr/opt/db2_08_01

- En todos los demás sistemas operativos basados en UNIX, el directorio de instalación es /opt/IBM/db2/V8.1

-a AuthType
Representa el tipo de autenticación para la instancia. AuthType puede ser uno de los siguientes: SERVER, CLIENT, DCS, SERVER_ENCRYPT, DCS_ENCRYPT. SERVER es el valor por omisión. Este parámetro es opcional.

-u FencedID
Representa el nombre del usuario bajo el que las funciones definidas por el usuario delimitado (las UDF) y los procedimientos almacenados delimitados se ejecutarán. Este distintivo no es necesario si está creando una instancia en un cliente DB2. Especifique el nombre del usuario delimitado que ha creado.

InstName
Representa el nombre de instancia. El nombre de la instancia debe ser el mismo que el nombre del usuario que es propietario de la instancia. Especifique el nombre del usuario que es propietario de la instancia que ha creado. La instancia se creará en el directorio inicial del usuario que es propietario de la instancia.

Por ejemplo, si está utilizando autenticación de servidor, el usuario delimitado es db2fenc1 y el usuario propietario de la instancia es db2inst1. Utilice el mandato siguiente para crear una instancia en un sistema AIX:

```
/usr/opt/db2_08_01/instance/db2icrt -a server -u db2fenc1 db2inst1
```

Después de crear una instancia es posible que desee configurar la notificación para la supervisión de salud. Esta tarea se puede realizar utilizando el Centro de salud o CLP. Consulte el manual *Suplemento de instalación y configuración de DB2* para obtener más información.

Verificación de la instalación de Spatial Extender

Esta tarea forma parte de la tarea más amplia de instalar y configurar Spatial Extender.

Después de instalar DB2 Spatial Extender, puede crear una base de datos y ejecutar el programa de verificación de la instalación para verificar que DB2 Spatial Extender está correctamente instalado y configurado.

Puede verificar la instalación utilizando el programa de ejemplo de DB2 Spatial Extender, runGseDemo. El programa runGseDemo está diseñado para poner de manifiesto problemas de la instalación. Durante la verificación de la instalación, el usuario puede recibir mensajes de error que le pueden ayudar a diagnosticar determinados problemas del sistema. La mayoría de los mensajes de error están causados por un pequeño número de problemas habituales. Para evitar estos errores, consulte el apartado "Requisitos".

Los pasos de verificación de esta sección se aplican a los siguientes sistemas operativos: Windows, AIX, HP-UX, Entornos operativos Solaris, Linux para Intel y Linux para S/390.

Requisitos previos:

Antes de ejecutar el programa runGseDemo:

- Compruebe que ha instalado el producto DB2 Spatial Extender en los entornos apropiados.
- Utilice una nueva base de datos que no tenga ninguna operación espacial asociada a ella.
- Para instalaciones UNIX (AIX, HP-UX, Entornos operativos Solaris, Linux para Intel y Linux para S/390), compruebe que ha establecido en entorno de instancias DB2 Spatial Extender. Consulte el manual *Suplemento de instalación y configuración de DB2* para obtener información acerca de cómo ejecutar el programa **db2ilist** para verificar las instancias. Es posible que necesite ejecutar el mandato **db2start** para iniciar la instancia de DB2..
- Aumente el valor del parámetro de configuración de la base de datos para el tamaño de la pila de aplicaciones. Para obtener detalles, consulte el apartado realizar la resolución de problemas de la instalación.

Procedimiento:

Para verificar la instalación:

1. Sólo para UNIX: inicie la sesión como propietario de la instancia.
2. Cree una base de datos.
   Abra la Ventana de mandatos de DB2 y entre:
   ```
   db2 create database mi_bd
   ```
   donde *mi_bd* es el nombre de la base de datos.
3. Localice el programa de verificación de la instalación.
   a. Para los sistemas operativos UNIX, entre:
      ```
      cd $HOME/sqllib/samples/spatial
      ```
      donde *$INICIO* es el directorio inicio del propietario de la instancia.
   b. Para Windows, entre:
      ```
      cd c:\Archivos de programa\IBM\sqllib\samples\spatial
      ```
      donde *c:\Archivos de programa\IBM\sqllib* es el directorio en el que ha instalado DB2 Spatial Extender.
4. Ejecute el programa de verificación de la instalación.
   En la línea de mandatos de DB2, entre el mandato **runGseDemo**. Por ejemplo, entre:
   ```
   runGseDemo mi_bd ID_usuario contraseña
   ```
   donde *mi_bd* es el nombre de la base de datos.

Si recibe mensajes de error durante el proceso de verificación, necesita realizar la resolución de problemas de la instalación.

Tareas relacionadas:

- "Resolución de problemas de instalación" en la página 41
- "Instalación de DB2 Spatial Extender para Windows" en la página 27
- "Instalación de DB2 Spatial Extender para AIX" en la página 29
- "Instalación de DB2 Spatial Extender para HP-UX" en la página 31
- "Instalación de DB2 Spatial Extender para el Entorno operativo Solaris" en la página 33
- "Instalación de DB2 Spatial Extender para Linux" en la página 35

Resolución de problemas de instalación

Cuando ejecuta el programa de ejemplo, runGseDemo, para verificar que Spatial Extender se ha instalado correctamente, es posible que se encuentre con los siguientes errores:

La base de datos ya está habilitada para operaciones espaciales

Compruebe que la base de datos para la que está verificando la instalación es nueva y no tiene operaciones espaciales asociadas; si las tiene, el programa de ejemplo no se ejecutará correctamente.

Recibirá el siguiente mensaje de error si la base de datos para la cual está ejecutando el programa de ejemplo ya está habilitada para operaciones espaciales:

```
Habilitándose la base de datos logtst...
Salida de ENABLE_DB:
Código de retorno = -14
Texto del mensaje de terminación =
GSE0014E  La base de datos ya se ha habilitado para operaciones espaciales.
```

Para corregir este problema, descarte la base de datos y repita los pasos del apartado Verificación de la instalación de Spatial Extender.

Valor del parámetro de configuración del gestor de la base de datos para el tamaño de la pila de aplicaciones

Si APPLHEAPSZ no está configurado con el valor adecuado, obtendrá este mensaje de error al habilitar la base de datos para operaciones espaciales:

```
GSE0213N  Una operación de
vinculación no ha sido satisfactoria.
SQLERROR = "SQL0001N La vinculación o precompilación
no se ha completado satisfactoriamente.
SQLSTATE=00000".SQLSTATE=57011
```

Para aumentar el valor del parámetro de configuración de la base de datos correspondiente al tamaño de la pila de aplicaciones, escriba:

```
db2 update db cfg for nombre_base_datos using APPLHEAPSZ 2048
```

Si 2048 no es apropiado, aumente el parámetro APPLHEAPSZ en incrementos de 256.

Consideraciones posteriores a la instalación

Después de instalar Spatial Extender, tenga en cuenta las acciones siguientes:

- Bajar ArcExplorer
- Acceder a los datos de referencia del geocodificador

Bajada de ArcExplorer for DB2

IBM proporciona un navegador, creado por ESRI (Environmental Systems Research Institute) para IBM, que puede producir directamente resultados visuales para consultas de datos de DB2 Spatial Extender sin necesidad de un servidor de datos intermedio. Este navegador es ArcExplorer for DB2. Puede bajar una copia gratuita de ArcExplorer for DB2 desde el sitio Web de Spatial Extender de IBM en la siguiente ubicación:

http://www.ibm.com/software/data/spatial/

Para obtener más información sobre la instalación y utilización de ArcExplorer for DB2, consulte el apartado *Utilización de ArcExplorer*, que también está disponible como parte del producto ArcExplorer for DB2 que puede bajar desde el sitio Web de DB2 Spatial Extender.

Importante: DB2 Universal Database Versión 8.1 se proporciona con Software Developer Kit (SDK) de IBM para Java Versión 1.3.1. Cuando instale ArcExplorer for DB2, colóquelo en un directorio diferente del de DB2. Recuerde definir la variable de entorno CLASSPATH.

Acceso a los datos de referencia del geocodificador

Los datos de referencia del geocodificador del CD de Datos de referencia del geocodificador de DB2 Spatial Extender se crean específicamente para que funcionen con un geocodificador proporcionado por Spatial Extender. Se compone de datos de mapas básicos para la red de calles de los EE.UU. El geocodificador DB2SE_USA utiliza estos datos para determinar las coordenadas de direcciones en una base de datos habilitada de forma espacial. Colectivamente, estos datos del mapa básico se denominan *datos de referencia*. El geocodificador DB2SE_USA utiliza datos de direcciones (no espaciales) de la base de datos, los compara y asocia con los datos de referencia y los convierte en coordenadas que pueden ser almacenadas por DB2 Spatial Extender. Este proceso se denomina *geocodificación*.

Los datos de referencia proporcionados en el CD incluyen el archivo EDGELocator.loc. El geocodificador DB2SE_USA utiliza el archivo EDGELocator.loc para localizar datos de referencia determinados. Por ejemplo, si está geocodificando direcciones de California, Kentucky y Oregón, el geocodificador DB2SE_USA utiliza el archivo localizador del CD para determinar las ubicaciones de las direcciones.

Procedimiento:

Puede acceder directamente a los datos del geocodificador desde el CD puede copiar los datos en el disco duro. Para copiar los archivos de datos del geocodificador del CD en el entorno de servidor de DB2 Spatial Extender, siga los pasos descritos en esta sección.

Para sistemas operativos UNIX:

1. Monte el CD. Para obtener información sobre cómo montar un CD, consulte el manual *DB2 para UNIX Guía rápida de iniciación*.
2. Inicie una sesión como usuario con autorización root en la máquina servidor de destino.
3. Escriba uno de los mandatos siguientes:
   - Para AIX:
     ```
     cp /cdrom/db2/* /usr/opt/db2_08_01/gse/refdata/
     ```
   - Para todas las demás plataformas UNIX:
     ```
     cp /cdrom/db2/* /0pt/IBM/db2/V8.1/gse/refdata/
     ```

   Nota: Puede copiar los archivos de datos del geocodificador en cualquier directorio normal de la unidad local. Si selecciona copiar los archivos en un directorio que especifique, debe modificar el archivo localizador para que indique la nueva ubicación.
4. Finalice la sesión.

Para Windows, puede utilizar la ventana de mandatos o el Explorador de Windows.

Para acceder a los datos del geocodificador a través de la ventana de mandatos:

1. Pulse **Inicio** –>**Programa** –> **IBM DB2** –> **Ventana de mandatos**.
2. Escriba el siguiente mandato:

   ```
   copy d:\db2\* %db2path%\gse\refdata
   ```

   Donde *d:* es la letra que corresponde a la unidad de CD.

   Nota: Puede copiar los archivos de datos del geocodificador en cualquier directorio normal de la unidad local. Si selecciona copiar los archivos en un directorio que especifique, debe modificar el archivo localizador para que indique la nueva ubicación.

Para acceder a los datos del geocodificador a través del Explorador de Windows:

Copie todos los archivos de *d:*\db2 en *c:\Archivos de programa\IBM\sqllib*\gse\refdata, donde *d:* es la unidad de CD y *c:\Archivos de programa\IBM\sqllib* es el directorio donde está instalado DB2.

Tareas relacionadas:

- "Instalación y configuración de Spatial Extender" en la página 25

CD para datos y mapas de DB2 Spatial Extender

DB2 Spatial Extender se proporciona con siete CD de datos y mapas.

La información sobre datos y mapas, etiquetada como Datos y mapas de DB2 Spatial Extender,1 – 7, se proporciona en siete CD. La tabla siguiente proporciona un resumen de los datos que contiene cada CD.

Tabla 2. Información del CD de datos y mapas

CD de datos y mapas	Resumen de los tipos de datos de mapa
CD 1	Europa y mundo
CD 2	Canadá, Méjico y Estados Unidos
CD 3	Estados Unidos
CD 4	Estados Unidos (región occidental)
CD 5	Estados Unidos (región central)
CD 6	Estados Unidos (región oriental)
CD 7	Estados Unidos (región meridional)

Para obtener una descripción detallada de los datos proporcionados por ESRI, consulte el archivo de ayuda de ESRI, esridata.hlp, situado en el CD de Datos y mapas de DB2 Spatial Extender.

- Para Windows, vea el archivo de ayuda ubicado en *x:* esridata.hlp, donde *x:* es la unidad de CD.
- Para sistemas operativos UNIX, vea o imprima el archivo de ayuda situado en el CD en */cdrom/esridata.hlp* , donde */cdrom* es el punto de montaje.

Tareas relacionadas:

• "Configuración y habilitación de DB2 Geodetic Extender" en la página 173

Capítulo 5. Migración del entorno de Spatial Extender a DB2 Universal Database Versión 8

Esta sección explica cómo migrar DB2 Spatial Extender desde la Versión 8.1 a la Versión 8.2. También explica cómo utilizar el programa de utilidad de migración para migrar desde un entorno de 32 bits a un entorno de 64 bits.

Migración de una base de datos habilitada para operaciones espaciales

Si ha estado utilizando DB2 Spatial Extender Versión 8.1, debe completar los siguientes pasos antes de utilizar una base de datos existente habilitada para operaciones espaciales con DB2 Spatial Extender Versión 8.2 o DB2 Geodetic Extender Versión 8.2. Este tema describe los pasos necesarios que hay que llevar a cabo para migrar bases de datos habilitadas para operaciones espaciales desde una versión anterior de DB2 Spatial Extender.

Requisitos previos:

Antes de empezar el proceso de migración:

- Finalice todas las conexiones a la base de datos antes de ejecutar el programa de utilidad de migración.
- Asegúrese de que es sistema se ajuste a los requisitos de instalación para DB2 Spatial Extender Versión 8.2.
- Para realizar una copia de seguridad de una base de datos, debe tener autorización SYSADM, SYSCTRL o SYSMAINT para la base de datos.
- Para migrar una base de datos, debe tener autorización SYSADM.

Procedimiento:

Para migrar el entorno de DB2 Spatial Extender:

1. Realice una copia de seguridad de la base de datos de la Versión 8.1. Para obtener información acerca de cómo realizar una copia de seguridad de la base de datos, consulte el manual *Suplemento de instalación y configuración de DB2*.
2. Instale DB2 Universal Database Versión 8.2 y DB2 Spatial Extender Versión 8.2.
3. Migre la instancia de DB2 y las bases de datos de la Versión 8.1 a la Versión 8.2. Para obtener más información sobre cómo migrar la instancia de DB2 y las bases de datos, consulte el manual *Suplemento de instalación y configuración de DB2*.
4. Migre una base de datos habilitada para operaciones espaciales de la Versión 8.1 a la Versión 8.2 utilizando el programa de utilidad de migración de Spatial Extender.
   a. En un indicador de mandatos del sistema operativos, utilice el mandato **db2se migrate_v82** para migrar la base de datos.

      ```
      db2se migrate_v82 nombre_basedatos userId ID_usuario PW contraseña
      ```

      Para conocer la sintaxis de este mandato, consulte el apartado “El mandato db2se migrate_v82” en la página 47.

Mensajes de migración

Si la migración es satisfactoria, se visualiza el siguiente mensaje:

```
GSE0000I La operación se ha completado satisfactoriamente
```

Si la migración no es satisfactoria, se visualiza el siguiente mensaje:

```
GSE9002N Se ha producido un error durante un intento de llevar a cabo
la migración de la base de datos de Spatial Extender.
```

Nota: Es posible que se produzcan los siguientes errores durante la migración:

- La base de datos no está actualmente habilitada para operaciones espaciales.
- La base de datos no es una base de datos habilitada para operaciones espaciales de la Versión 8.1.
- La base de datos ya es una base de datos habilitada para operaciones espaciales de la Versión 8.2.
- El nombre de la base de datos no es válido.
- Existen otras conexiones a la base de datos. No se puede ejecutar.
- El catálogo espacial no es consistente.
- El usuario no está autorizado.
- La contraseña no es válida.
- No se han podido migrar algunos objetos de usuario.

No olvide comprobar el archivo de mensajes para obtener información detallada sobre cualquier error que reciba. El archivo de mensajes también contiene información útil, como, por ejemplo, índices, vistas y la configuración de geocodificación que se ha migrado.

Tareas relacionadas:

- "Copia de seguridad de bases de datos antes de la migración de DB2" en el manual *Guía rápida de iniciación para servidores DB2*
- "Migración de bases de datos" en el manual *Guía rápida de iniciación para servidores DB2*
- "Migración de instancias (UNIX)" en el manual *Guía rápida de iniciación para servidores DB2*
- "Migración de DB2 UDB (Windows)" en el manual *Guía rápida de iniciación para servidores DB2*
- "Migración de DB2 UDB (UNIX)" en el manual *Guía rápida de iniciación para servidores DB2*
- "Migración de DB2 Personal Edition (Windows)" en el manual *Guía rápida de iniciación para DB2 Personal Edition*
- "Migración de DB2 Personal Edition (Linux)" en el manual *Guía rápida de iniciación para DB2 Personal Edition*
- "Migración de bases de datos en DB2 Personal Edition (Windows)" en el manual *Guía rápida de iniciación para DB2 Personal Edition*
- "Migración de instancias y bases de datos en DB2 Personal Edition (Linux)" en el manual *Guía rápida de iniciación para DB2 Personal Edition*

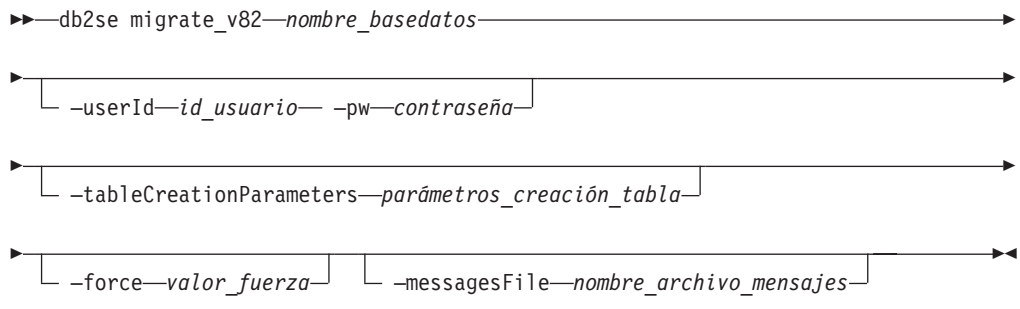
Información relacionada:

- "El mandato db2se migrate_v82" en la página 47

El mandato db2se migrate_v82

Utilice el mandato **db2se migrate_v82** para migrar una base de datos habilitada para operaciones espaciales de la Versión 8.1 a la Versión 8.2.

Sintaxis:



Donde:

-nombre_basedatos
: Nombre de la base de datos que desea migrar.

-ID_usuario
: ID de usuario de la base de datos que tiene la autorización SYSADM o DBADM sobre la base de datos que se está migrando.

***-**contraseña*
: Contraseña de usuario.

-nombre_archivo_mensajes
: Nombre del archivo que contiene el informe de las acciones de migración. El nombre de archivo que especifique debe ser un nombre de archivo calificado al completo en el servidor.

Tareas relacionadas:

- "Configuración y habilitación de DB2 Geodetic Extender" en la página 173
- "Instalación y configuración de Spatial Extender" en la página 25
- "Habilitación de una base de datos para operaciones espaciales" en la página 54

Migración

Capítulo 6. Configuración de una base de datos

Este capítulo trata sobre cómo configurar una base de datos para albergar datos espaciales.

Configuración de una base de datos para alojar datos espaciales

DB2 Spatial Extender, que se ejecuta en un entorno de DB2 Universal Database, funciona con la mayoría de valores por omisión de configuración de DB2. Sin embargo, algunos parámetros de configuración afectan a las operaciones espaciales. Debe ajustar estos parámetros de forma que las aplicaciones espaciales funcionen lo más eficazmente posible. En algunos casos, es necesario seleccionar un valor distinto del valor por omisión para las operaciones espaciales. En otros casos, hacer esto es recomendable, en función de las aplicaciones y del entorno general de DB2. Este tema identifica los parámetros de configuración de DB2 que influencian las operaciones de DB2 Spatial Extender.

Las secciones siguientes explican cómo ajustar el gestor de bases de datos de DB2 y los parámetros de configuración de la base de datos que afectan a las operaciones de DB2 Spatial Extender.

Ajuste de los parámetros de configuración de la base de datos

Algunos parámetros de configuración de la base de datos afectan a aplicaciones espaciales. Para modificar cualquier parámetro de configuración de la base de datos, debe conectarse a la base de datos. Cuando modifique los valores de estos parámetros para una base de datos, el cambio afectará sólo a dicha base de datos. Las secciones siguientes explican cómo ajustar los parámetros para aplicaciones espaciales:

- "Ajuste de las características de anotaciones cronológicas de transacción"
- "Ajuste del tamaño de la pila de aplicaciones" en la página 51
- "Ajuste del tamaño de la pila de control de aplicaciones" en la página 52

Ajuste de las características de anotaciones cronológicas de transacción

Antes de habilitar una base de datos para operaciones espaciales, asegúrese de tener suficiente capacidad para el registro de transacciones. Los valores por omisión para los parámetros de configuración de las anotaciones cronológicas de transacción no proporcionan suficiente capacidad de anotaciones cronológicas de transacción si piensa realizar cualquiera de las acciones siguientes:

- Habilitar una base de datos para operaciones espaciales en un entorno Windows
- Utilizar el procedimiento almacenado ST_import_shape para importar desde archivos de formas
- Utilizar la geocodificación con un ámbito de confirmación amplio
- Ejecutar transacciones simultáneas

Si piensa realizar alguna de estas acciones ahora o en el futuro, es necesario aumentar la capacidad de las anotaciones cronológicas de transacción para la base de datos aumentando uno o varios parámetros de configuración de las anotaciones

cronológicas de transacción. Si éste no es su caso, puede utilizar las características por omisión. En este caso, siga con el apartado "Ajuste del tamaño de la pila de aplicaciones" en la página 51.

Recomendación: Consulte la tabla siguiente para ver los valores mínimos recomendados para los tres parámetros de configuración de anotaciones cronológicas de transacción.

Tabla 3. Valores mínimos recomendados para los parámetros de configuración de transacción

Parámetro	Descripción	Valor por omisión	Valor mínimo recomendado
LOGFILSZ	Especifica el tamaño del archivo de anotaciones cronológicas como un número de bloques de 4 KB	1000	1000
LOGPRIMARY	Especifica cuántos archivos de anotaciones cronológicas primarios se van a preasignar a los archivos de anotaciones cronológicas de recuperación	3	10
LOGSECOND	Especifica el número de archivos de anotaciones cronológicas secundarios	2	2

Si la capacidad de las anotaciones cronológicas de transacción no es la adecuada, se emitirá el siguiente mensaje de error cuando intente habilitar una base de datos para operaciones espaciales:

```
GSE0010N No hay suficiente espacio de anotaciones cronológicas disponible en DB2.
```

Procedimiento:

Para aumentar el valor de uno o varios parámetros de configuración:

1. Busque el valor actual para los parámetros LOGFILSIZ, LOGPRIMARY y LOGSECOND revisando los datos de salida del mandato GET DATABASE CONFIGURATION o desde la ventana **Configurar base de datos** del Centro de control de DB2.
2. Decida si cambiar uno, dos o tres valores de la forma indicada en la tabla de más arriba.
3. Cambie cada valor que desee modificar. Puede cambiar los valores emitiendo uno o varios de los siguientes mandatos, donde *nombre_bd* identifica la base de datos:

   ```
   UPDATE DATABASE CONFIGURATION FOR nombre_bd USING LOGFILSZ 1000
   ```

```
UPDATE DATABASE CONFIGURATION FOR nombre_bd USING LOGPRIMARY 10

UPDATE DATABASE CONFIGURATION FOR nombre_bd USING LOGSECOND 2
```

Si el único parámetro que cambia es LOGSECOND, el cambio tiene efecto inmediato. En este caso, siga con el apartado "Ajuste del tamaño de la pila de aplicaciones".

4. Si cambia el parámetro LOGFILSIZ o LOGPRIMARY, o ambos:
   a. Desconecte todas las aplicaciones de la base de datos.
   b. Si la base de datos se ha activado explícitamente, desactive la base de datos.

   Los cambios en los parámetros LOGFILSIZ o LOGPRIMARY, o en ambos, serán efectivos la próxima vez que se active la base de datos o que se establezca una conexión con la base de datos.

Ajuste del tamaño de la pila de aplicaciones

Utilice el parámetro de configuración de la base de datos APPLHEAPSZ para especificar el tamaño de la pila de aplicaciones (en número de páginas de 4 KB). Este parámetro define el número de páginas de memoria privada que están disponibles para que las utilice el gestor de la base de datos en nombre de un agente o subagente específico. La pila se asigna cuando se inicializa un agente o subagente para una aplicación. La cantidad asignada es la cantidad mínima que es necesaria para procesar la petición dirigida al agente o subagente. Debido a que el agente o subagente necesita más espacio de pila para procesar sentencias de SQL mayores, el gestor de bases de datos asigna memoria según sea necesario, hasta el máximo especificado en este parámetro. La pila de aplicaciones se asigna a partir de la memoria privada del agente.

El valor por omisión para el parámetro APPLHEAPSZ es 128 (páginas de 4 KB). Cuando ejecuta el procedimiento almacenado ST_enable_db, este valor debe ser como mínimo 2048.

Recomendación: para la mayoría de las aplicaciones de DB2 Spatial Extender, especialmente aquéllas que importan o exportan archivos de formas, utilice un valor para el parámetro APPLHEAPSZ que sea cómo mínimo 2048.

Si APPLHEAPSZ está establecido en un valor inadecuado, se emitirá el siguiente mensaje de error cuando intente habilitar una base de datos para operaciones espaciales:

```
GSE0009N No hay espacio suficiente disponible en la pila de
aplicaciones de DB2.

GSE0213N Ha fallado una operación de vinculación. SQLERROR = "SQL0001N
La vinculación o la precompilación no ha finalizado satisfactoriamente.
SQLSTATE=00000".
```

Procedimiento:

Para cambiar el tamaño de la pila de aplicaciones:

1. Busque el valor actual para el parámetro APPLHEAPSZ revisando la salida del mandato GET DATABASE CONFIGURATION o desde la ventana **Configurar base de datos** del Centro de control de DB2.
2. Cambie el valor al valor recomendado de 2048 o a un valor mayor. Puede cambiar el valor a 2048 emitiendo el siguiente mandato, donde *nombre_bd* identifica la base de datos:

   ```
   UPDATE DATABASE CONFIGURATION FOR nombre_bd USING APPLHEAPSZ 2048
   ```

3. Desconecte todas las aplicaciones de la base de datos.
4. Si la base de datos se ha activado explícitamente, desactive la base de datos.

El cambio será efectivo la próxima vez que se active la base de datos o que se establezca una conexión con la base de datos.

Ajuste del tamaño de la pila de control de aplicaciones

Todas las aplicaciones de DB2 Spatial Extender, especialmente las que importan de archivos de formas o exportan a éstos, se pueden beneficiar de la utilización del valor recomendado para el tamaño de la pila de control de aplicaciones. Especifique este valor con el parámetro APP_CTL_HEAP_SZ. Este parámetro especifica el tamaño máximo, en páginas de 4 KB, para la memoria compartida de control de aplicaciones. Las pilas de control de aplicaciones se asignan a partir de esta memoria compartida. Se asigna una pila de control de aplicaciones para cada aplicación de la base de datos donde la aplicación está activa (o, en el caso de un sistema de base de datos particionada, en cada partición de base de datos donde la aplicación está activa). La pila se asigna durante el proceso de conexión por el primer agente que recibe una petición para la aplicación de la base de datos (o de la partición de base de datos). La pila se utiliza para compartir información entre agentes que trabajan para la misma aplicación. (En un entorno de base de datos particionada, el compartimiento se realiza a nivel de partición de base de datos; no hay compartimiento entre particiones de base de datos.) El valor por omisión para el parámetro APP_CTL_HEAP_SZ es 128.

Recomendación: Para la mayoría de aplicaciones DB2 Spatial Extender, utilice el valor del parámetro APP_CTL_HEAP_SZ de cómo mínimo 1024 (páginas de 4 KB).

Si APP_CTL_HEAP_SZ está establecido en un valor inadecuado, se emitirá el siguiente mensaje de error al importar datos a la base de datos desde archivos de formas:

```
GSE0214N Ha fallado una sentencia INSERT. SQLERROR = "SQL0973N No hay suficiente
almacenamiento disponible en la pila "APP_CTL_HEAP" para procesar la sentencia.
```

Procedimiento:

Para cambiar el tamaño de la pila de control de aplicaciones:

1. Busque el valor actual del parámetro APP_CTL_HEAP_SZ examinando los datos de salida del mandato GET DATABASE CONFIGURATION o en la ventana **Configurar base de datos** del Centro de control de DB2.
2. Cambie el valor al valor recomendado de 1024 (páginas de 4 KB) o a un valor mayor. Puede emitir el siguiente mandato, donde *nombre_bd* identifica la base de datos:

   ```
   UPDATE DATABASE CONFIGURATION FOR nombre_bd USING APP_CTL_HEAP_SZ 1024
   ```
3. Desconecte todas las aplicaciones de la base de datos.
4. Si la base de datos se ha activado explícitamente, desactive la base de datos.

El cambio será efectivo la próxima vez que se active la base de datos o que se establezca una conexión con la base de datos.

Capítulo 7. Configuración de recursos espaciales para una base de datos

Después de configurar la base de datos para alojar datos espaciales, está preparado para proporcionar a la base de datos recursos que necesitará al crear y gestionar columnas espaciales y al analizar datos espaciales. Estos recursos incluyen:

- Objetos proporcionados por Spatial Extender para dar soporte a operaciones espaciales; tales como, procedimientos almacenados para administrar una base de datos, tipos de datos espaciales y programas de utilidad para geocodificar e importar datos espaciales.
- Datos de referencia: Rangos de direcciones que DB2SE_USA_GEOCODER utiliza para convertir direcciones individuales en coordenadas.
- Cualquier codificador que los usuarios o los proveedores proporcionen.

Este capítulo describe estos recursos y presenta las tareas mediante las cuales el usuario los hace disponibles: habilitación de la base de datos para operaciones espaciales, configuración del acceso a datos de referencia y registro de geocodificadores que no son los geocodificadores por omisión.

Cómo configurar recursos en la base de datos

La primera tarea que realiza después de configurar la base de datos para alojar datos espaciales es hacer que la base de datos pueda dar soporte a operaciones espaciales—operaciones como, por ejemplo, llenar tablas con datos espaciales y procesar consultas espaciales. Esta tarea implica cargar la base de datos con ciertos recursos proporcionados por DB2 Spatial Extender. Esta sección describe estos recursos y describe brevemente la tarea.

Inventario de recursos proporcionados para la base de datos

Para habilitar una base de datos para operaciones espaciales, DB2® Spatial Extender proporciona a la base de datos los siguientes recursos:

- Procedimientos almacenados. Cuando el usuario solicita una operación espacial—por ejemplo, cuando emite un mandato para importar datos espaciales—DB2 Spatial Extender invoca a uno de estos procedimientos almacenados para realizar la operación.
- Tipos de datos espaciales. Debe asignar un tipo de datos espaciales a cada columna de tabla o de vista en la que se van a almacenar datos espaciales.
- Catálogo de DB2 Spatial Extender. Determinadas operaciones dependen de este catálogo. Por ejemplo, para poder acceder a la columna espacial desde las herramientas de visualización, la herramienta podría necesitar que la columna espacial esté registrada en el catálogo.
- Un índice reticular espacial. Le permite definir índices reticulares sobre columnas espaciales.
- Funciones espaciales. Sirven para trabajar con datos espaciales de diversas formas; por ejemplo, para determinar las relaciones entre geometrías y para generar más datos espaciales.
- Definiciones de sistemas de coordenadas.
- Sistemas de referencia espacial por omisión.

- Dos esquemas: DB2GSE y ST_INFORMTN_SCHEMA. DB2GSE contiene los objetos que se acaban de listar: procedimientos almacenados, tipos de datos espaciales, el catálogo de DB2 Spatial Extender, etc. Las vistas del catálogo están disponibles también en ST_INFORMTN_SCHEMA para cumplir el estándar SQL/MM.

Habilitación de una base de datos para operaciones espaciales

La tarea de que DB2 Spatial Extender proporcione una base de datos con recursos para crear columnas espaciales y manipular datos espaciales se suele denominar "habilitación de la base de datos para operaciones espaciales".

Requisito necesario:

Antes de habilitar una base de datos para operaciones espaciales, el ID de usuario debe tener autorización SYSADM o DBADM sobre la base de datos.

Restricciones:

Sólo puede utilizar tipos de datos creados por el mandato enable_db.

Procedimiento:

Puede habilitar una base de datos para operaciones espaciales de cualquiera de estas maneras:

- Utilice la ventana Habilitar base de datos desde la opción de menú DB2 Spatial Extender. La opción de menú está disponible desde el objeto de base de datos del Centro de control de DB2.
- Emita el mandato **db2se enable_db**.
- Ejecute una aplicación que llame al procedimiento almacenado db2gse.ST_enable_db.

Nota:

Puede seleccionar explícitamente el espacio de tabla donde desea que resida el catálogo de DB2 Spatial Extender. Si no lo hace, DB2 utilizará el espacio de tabla por omisión.

Conceptos relacionados:

- "Inventario de recursos proporcionados para la base de datos" en la página 53

Tareas relacionadas:

- "Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 141

Información relacionada:

- "Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos" en la página 131
- "ST_enable_db" en la página 261

Cómo trabajar con datos de referencia

Esta sección explica qué son los datos de referencia e indica lo que el usuario necesita hacer para poder acceder a los mismos.

Datos de referencia

Los datos de referencia son rangos de direcciones que DB2SE_USA_GEOCODER utiliza para convertir direcciones individuales en coordenadas. Estos datos consisten en rangos de las direcciones más recientes que ha recopilado la Oficina del censo de Estados Unidos. Cuando DB2SE_USA_GEOCODER lee una dirección en la base de datos, busca los datos de referencia para:

- Los nombres de ciertas calles dentro de la zona designada por el código postal de la dirección. El geocodificador busca nombres que coincidan con el nombre de la calle contenida en la dirección en un grado especificado, o en un grado mayor que el especificado; por ejemplo, 80 por ciento o mayor.
- El rango de direcciones que corresponde con el número de la dirección.

Si se encuentra una coincidencia que no tiene el grado de concordancia solicitado, el geocodificador devuelve las coordenadas de la dirección que ha leído. Si no se encuentra una coincidencia o no tiene el grado de concordancia solicitado, el geocodificador devuelve un valor nulo.

Puede utilizar un archivo de configuración avanzada, llamado *archivo localizador*, para actuar más sobre el proceso realizado por el geocodificador, DB2SE_USA_GEOCODER. Normalmente no es necesario modificar la configuración por omisión proporcionada por DB2® Spatial Extender en ese archivo.

Configuración del acceso a los datos de referencia

Los datos de referencia para DB2SE_USA_GEOCODER están en uno de los CD en los que se proporciona Spatial Extender. Esta sección describe cómo preparar el acceso a los mismos.

Procedimiento:

Para preparar el acceso a los datos de referencia del geocodificador por omisión:

1. Decida si desea conservar los datos de referencia en el CD o bien almacenarlos en el disco duro. Si los conserva en el CD, se ahorrará el espacio (unos 700 MB aproximadamente) que ocuparían en el disco duro. Si los almacena en el disco duro podrá recuperarlos más rápidamente que si los conserva en el CD.
2. Si desea almacenar los datos de referencia en el disco duro:
   a. Verifique que el disco duro tiene espacio suficiente para contener los datos (aproximadamente 700 megabytes).
   b. Copie los datos en el disco duro. Para ver instrucciones, consulte el archivo README que acompaña a los datos de referencia.
   c. Determine si la copia ha sido satisfactoria: para verificar en UNIX si los datos se han cargado correctamente, búsquelos en el directorio $DB2INSTANCE/sqllib/gse/refdata/. Para comprobar en Windows NT si los datos se han cargado correctamente, búsquelos en el directorio %DB2PATH%\gse\refdata\.

3. Indique a DB2SE_USA_GEOCODER el nombre y la ubicación del archivo localizador y el mapa base. Haga esto configurando con los valores correctos los parámetros base_map y locator_file de DB2SE_USA_GEOCODER. Para ver más información, consulte con el administrador de la base de datos o póngase en contacto con el representante de IBM.

Registro de un geocodificador

DB2SE_USA_GEOCODER se registra automáticamente en DB2 Spatial Extender cuando se habilita una base de datos para operaciones espaciales. Antes de que se puedan utilizar otros geocodificadores, éstos también se deben registrar.

Requisito necesario:

Antes de poder registrar un geocodificador, el ID de usuario debe tener autoridad SYSADM o DBADM sobre la base de datos en la que se ubica el geocodificador.

Procedimiento:

Puede registrar un geocodificador de cualquiera de estas maneras:

- Regístrelo desde la ventana Registrar geocodificador del Centro de control de DB2.
- Emita el mandato **db2se register_gc**.
- Ejecute una aplicación que llame al procedimiento almacenado db2gse.ST_register_geocoder.

Conceptos relacionados:

- "Geocodificadores y geocodificación" en la página 95

Parte 3. Creación de proyectos que utilizan datos espaciales

Capítulo 8. Configuración de recursos espaciales para un proyecto

Después de habilitar la base de datos para operaciones espaciales, está preparado para crear proyectos que utilicen datos espaciales. Entre los recursos que cada proyecto necesita está un sistema de coordenadas al que se adecuan los datos espaciales y un sistema de referencia espacial que define la extensión del área geográfica a la que los datos hacen referencia. Este capítulo:

- Trata la naturaleza de los sistemas de coordenadas e indica cómo crearlos
- Explica qué son los sistemas de referencia espacial e indica como crearlos

Cómo utilizar los sistemas de coordenadas

Cuando planea un proyecto que utiliza datos espaciales, es necesario determinar si los datos se deben basar en uno de los sistemas de coordenadas que están registrados en el catálogo de Spatial Extender. Si ninguno de estos sistemas de coordenadas cumple los requisitos, puede crear uno que lo haga. Esta explicación explica el concepto de sistemas de coordenadas y presenta las tareas de seleccionar uno para utilizarlo y de crear uno nuevo.

Sistemas de coordenadas

Un sistema de coordenadas es un marco que define las ubicaciones relativas de las cosas en un área determinada; por ejemplo, un área de la superficie de la tierra o la superficie de la tierra en su totalidad. DB2® Spatial Extender da soporte a los siguientes tipos de sistemas de coordenadas para determinar la ubicación de un elemento geográfico:

Sistema de coordenadas geográficas
Un *sistema de coordenadas geográficas* es un sistema de referencia (consulte el apartado "Sistemas de referencia espacial" en la página 68) que utiliza una superficie esférica tridimensional para determinar ubicaciones en la tierra. Se puede hacer referencia a cualquier ubicación de la tierra mediante un punto con coordenadas de latitud y longitud basado unidades angulares de medida.

Sistema de coordenadas proyectadas
Un *sistema de coordenadas proyectadas* es una representación bidimensional plana de la tierra. Utilice coordenadas rectilíneas (cartesianas) basadas en unidades lineales de medida. Se basa en un modelo esférico (o esferoidal) de la tierra y sus coordenadas se relacionan con coordenadas geográficas mediante una transformación de proyección.

Conceptos relacionados:

- "Sistema de coordenadas geográficas" en la página 60
- "Sistemas de coordenadas proyectadas" en la página 65

Información relacionada:

- "Sistemas de coordenadas soportados" en la página 539

Sistema de coordenadas geográficas

Un *sistema de coordenadas geográficas* es un sistema de referencia (consulte el apartado "Sistemas de referencia espacial" en la página 68) que utiliza una superficie esférica tridimensional para determinar ubicaciones en la tierra. Se puede hacer referencia a cualquier ubicación de la tierra mediante un punto con coordenadas de longitud y latitud. Los valores de los puntos pueden tener las siguientes unidades de medida:

- Unidades lineales cuando el sistema de coordenadas geográficas tiene un identificador de sistemas de referencia espacial (SRID) que DB2® Geodetic Extender reconoce.
- Cualquiera de las siguientes unidades cuando el sistema de coordenadas geográficas tiene un SRID que DB2 Geodetic Extender no reconoce.
  - Grados decimales
  - Minutos decimales
  - Segundos decimales
  - Gradianes
  - Radianes

Para el rango de valores de estas unidades, consulte el apartado "Sistemas de coordenadas soportados" en la página 539.

Por ejemplo, la Figura 6 muestra un sistema de coordenadas geográficas en el que una ubicación se representa mediante las coordenadas de 80 grados este de longitud y de 55 grados norte de latitud.

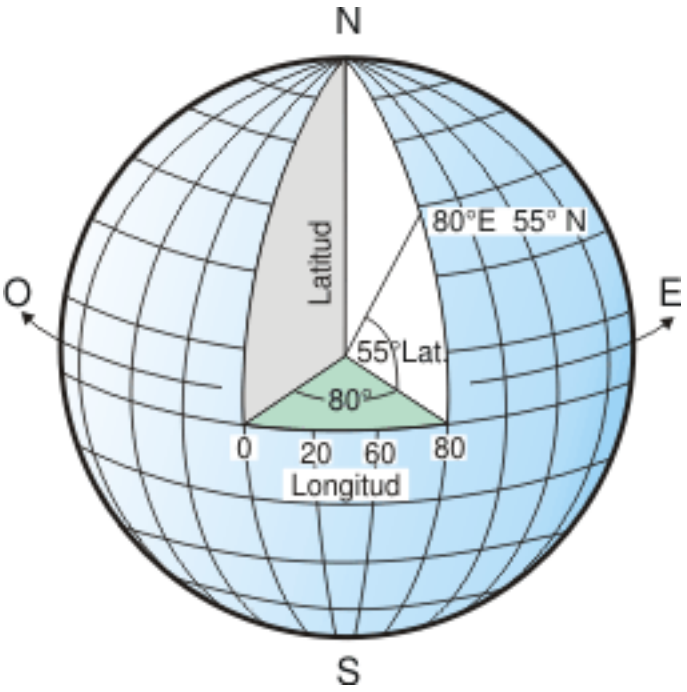


Figura 6. Un sistema de coordenadas geográficas

Las líneas que van del este al oeste tienen un valor de latitud constante y se denominan *paralelos*. Son equidistantes y paralelas entre ellas, y forman círculos concéntricos alrededor de la tierra. El *ecuador* es el círculo más largo y divide la tierra por la mitad. Es equidistante de cada uno de los polos y el valor de esta línea de latitud es de cero. Las ubicaciones situadas al norte del ecuador tienen

latitudes positivas comprendidas entre 0 y +90 grados, mientras que las ubicaciones situadas al sur del ecuador tienen latitudes negativas comprendidas entre 0 y -90 grados.

La Figura 7 ilustra las líneas de latitud.

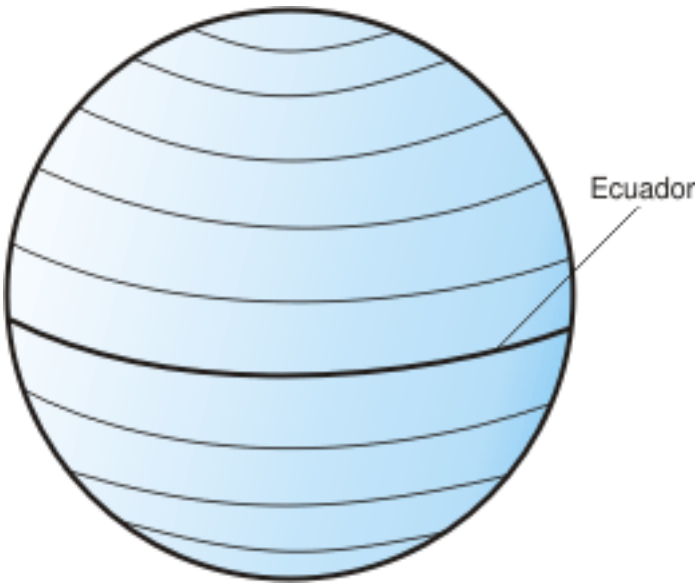


Figura 7. Líneas de latitud

Las líneas que van del norte al sur tienen un valor de longitud constante y se denominan *meridianos*. Forman círculos del mismo tamaño alrededor de la tierra y forman intersecciones en los polos. El *meridiano de origen* es la línea de longitud que define el origen (cero grados) de las coordenadas de longitud. Una de las ubicaciones del meridiano de origen utilizadas más habitualmente es la línea que pasa por Greenwich, Inglaterra. Sin embargo, existen otras líneas de longitud, como las que pasan por Berna, Bogotá y Paris, que también pueden utilizarse como meridiano de origen. Las ubicaciones situadas al este del meridiano de origen hasta su meridiano *antípoda* (la continuación del meridiano de origen en el otro lado del globo) tienen longitudes positivas comprendidas entre 0 y +180 grados. Las ubicaciones situadas al oeste del meridiano de origen tienen longitudes negativas comprendidas entre 0 y –180 grados.

La Figura 8 en la página 62 muestra las líneas de longitud.

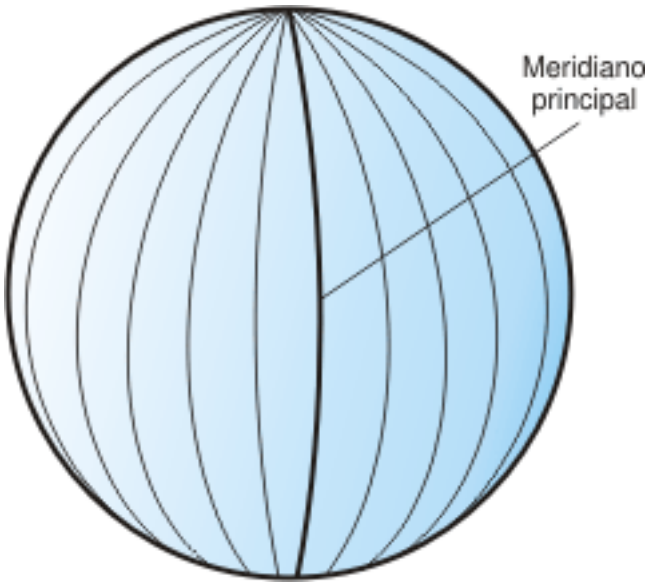


Figura 8. Líneas de longitud

Las líneas de latitud y longitud pueden recubrir el globo para formar una cuadrícula denominada *red geográfica*. El punto de origen de la red geográfica es (0,0), donde el ecuador y el meridiano de origen forman intersección. El ecuador es el único lugar de la red geográfica donde la distancia lineal correspondiente a un grado de latitud equivale aproximadamente a la de un grado de longitud. Debido a que las líneas de longitud convergen en los polos, la distancia entre dos meridianos es diferente en cada paralelo. Por lo tanto, a medida que nos acercamos a los polos, la distancia correspondiente a un grado de latitud será significativamente superior a la distancia correspondiente a un grado de longitud.

También es difícil determinar las longitudes de las líneas de latitud utilizando la red geográfica. Las líneas de latitud son círculos concéntricos que se hacen más pequeños cerca de los polos. Forman un único punto en los polos donde empiezan los meridianos. En el ecuador, un grado de longitud equivale aproximadamente a 111.321 kilómetros, mientras que a 60 grados de latitud, un grado de longitud equivale a sólo 55.802 km (esta aproximación se basa en el esferoide Clarke 1866). Por lo tanto, debido a que no existe una longitud uniforme de grados de latitud y longitud, la distancia entre los puntos no puede medirse con precisión utilizando unidades angulares de medida.

La Figura 9 en la página 63 muestra las diferentes dimensiones entre ubicaciones en la red geográfica.

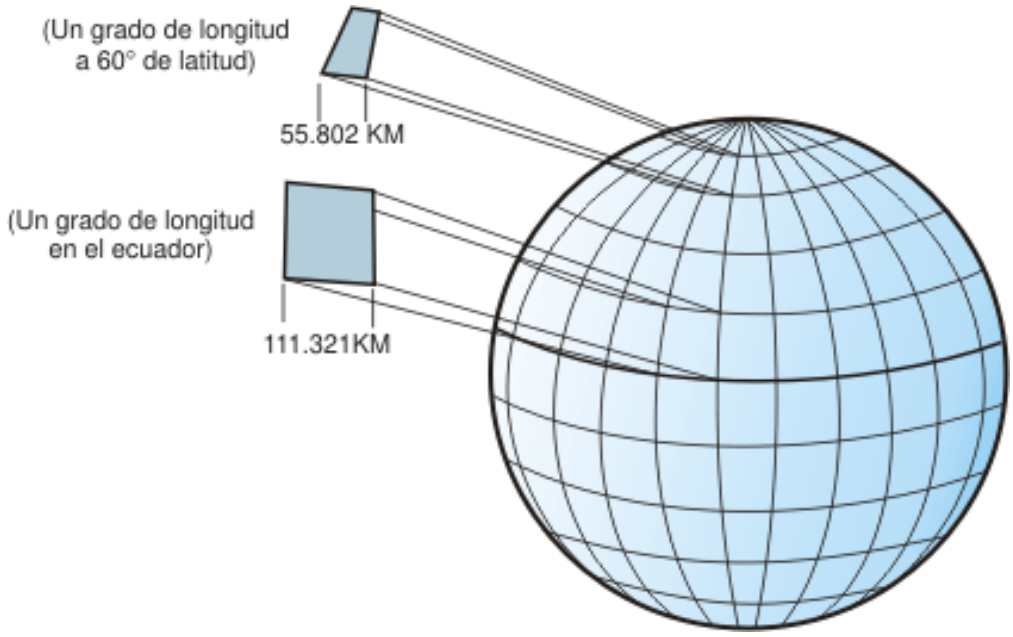


Figura 9. Diferentes dimensiones entre ubicaciones en la red geográfica

Un sistema de coordenadas se puede definir mediante una aproximación de esfera o esferoide de la forma de la tierra. Debido a que la tierra no es completamente redonda, un esferoide puede ayudar a mantener la precisión de un mapa, dependiendo de la ubicación en la tierra. Un *esferoide* es un elipsoide, que está basado en una elipse, mientras que una esfera está basada en un círculo.

La forma de la elipse está determinada por dos radios. El radio mayor se denomina eje semimayor y el radio menor se denomina eje semimenor. Un elipsoide es una forma tridimensional generada al girar una elipse alrededor de uno de sus ejes.

La Figura 10 en la página 64 muestra las aproximaciones de esfera y esferoide de la tierra y los ejes mayor y menor de una elipse.

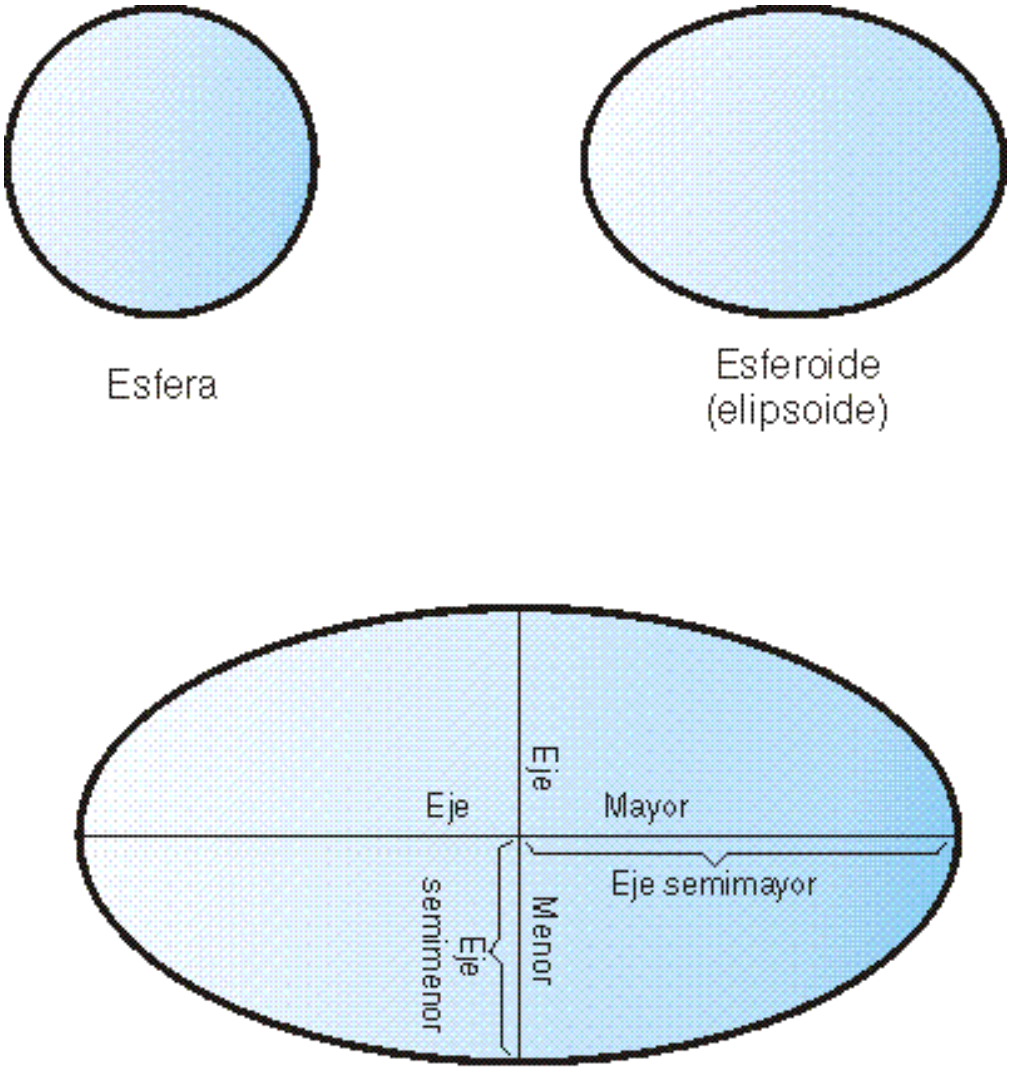


Figura 10. Aproximaciones de esfera y esferoide

Un *datum* es un conjunto de valores que define la posición del esferoide con relación al centro de la tierra. El datum proporciona un marco de referencia para medir ubicaciones y define el origen y la orientación de las líneas de latitud y longitud. Algunos sistemas son globales y pretenden proporcionar una buena precisión media en todo el mundo. Un datum local alinea su esferoide para que se ajuste con precisión a la superficie de la tierra en una zona determinada. Por lo tanto, las mediciones del sistema de coordenadas no serán precisas si se utilizan con un área distinta del área para la que está diseñado. Para obtener más información sobre elipsoides, consulte el apartado “Sistemas de coordenadas soportados” en la página 539.

La Figura 11 en la página 65 muestra cómo sistemas de referencia diferentes se alinean con la superficie de la tierra. El datum local, NAD27, se alinea más

estrechamente con la superficie de la tierra que el datum centrado en la tierra, WGS84, en esta ubicación particular.

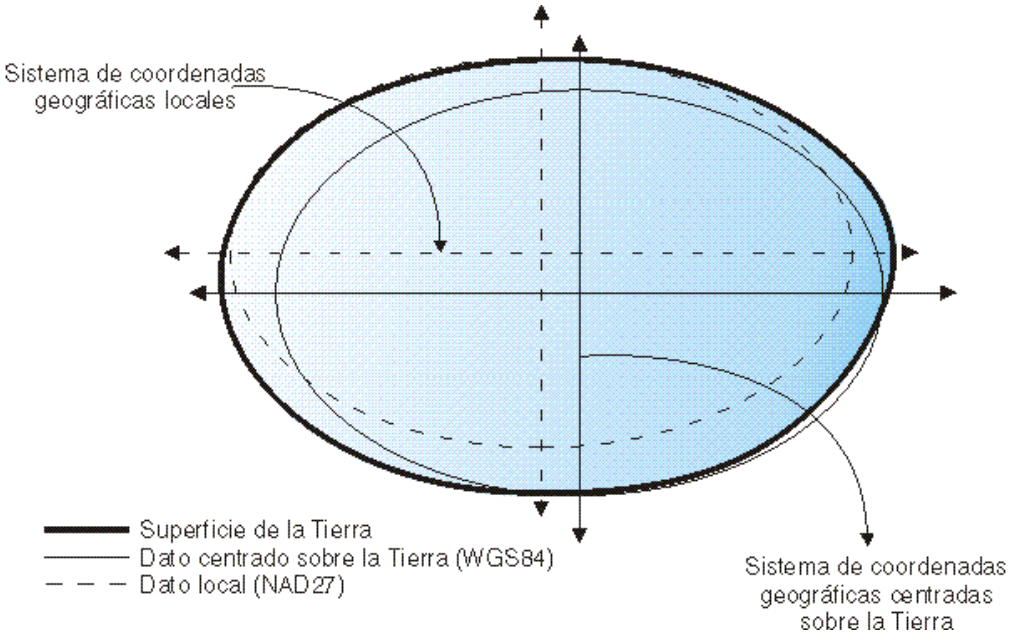


Figura 11. Alineaciones de sistemas de referencia

Siempre que cambie el datum, el sistema de coordenadas geográficas se modificará y los valores de coordenadas cambiarán. Por ejemplo, las coordenadas en DMS de un punto de control situado en Redlands, California, utilizando el datum NAD 1983 (North American Datum de 1983) son: "-117 12 57.75961 34 01 43.77884" Las coordenadas del mismo punto en el datum NAD 1927 (North American Datum de 1927) son: "-117 12 54.61539 34 01 43.72995" .

Sistemas de coordenadas proyectadas

Un *sistema de coordenadas proyectadas* es una representación plana y bidimensional de la tierra. Se basa en un sistema de coordenadas geográficas esféricas o esferoidales, pero utiliza unidades de medida lineales para las coordenadas, de forma que los cálculos de distancia y área se pueden realizar en términos de esas mismas unidades.

Las coordenadas de longitud y latitud se convierten en coordenadas x, y en la proyección plana. La coordenada x representa normalmente la orientación hacia el este de un punto y la coordenada y representa la orientación hacia el norte. La línea central que va de este a oeste se denomina eje x, y la línea central que va del norte al sur se denomina eje y.

La intersección de los ejes x e y es el origen y normalmente tiene la coordenada (0,0). Los valores situados por encima del eje x son positivos y los valores situados por debajo del eje x son negativos. Las líneas paralelas al eje x son equidistantes entre ellas. Los valores situados a la derecha del eje y son positivos y los valores situados a la izquierda del eje y son negativos. Las líneas paralelas al eje y son equidistantes.

Se utilizan fórmulas matemáticas para convertir un sistema de coordenadas geográficas tridimensionales en un sistema de coordenadas proyectadas planas

bidimensionales. La transformación se denomina *proyección cartográfica*. Las proyecciones cartográficas normalmente se clasifican según la superficie de proyección utilizada, como, por ejemplo, superficies cónicas, cilíndricas y planas. En función de la proyección utilizada, diferentes propiedades espaciales aparecerán distorsionadas. Las proyecciones están diseñadas para minimizar la distorsión de una o dos características de datos, pero es posible que la distancia, el área, la forma, la dirección o una combinación de estas propiedades no sean representaciones precisas de los datos de los que se está realizando el modelo. Existen varios tipos de proyecciones disponibles. Mientras que la mayoría de las proyecciones cartográficas intentan conservar cierta precisión de las propiedades espaciales, otras en su lugar intentan minimizar la distorsión general, como por ejemplo la proyección *Robinson*. Los tipos más habituales de proyecciones cartográficas incluyen los siguientes:

Proyecciones de áreas iguales
Estas proyecciones conservan el área de elementos específicos. Estas proyecciones distorsionan la forma, el ángulo y la escala. La proyección *cónica de áreas iguales Albers* es un ejemplo de una proyección de áreas iguales.

Proyecciones conformes
Estas proyecciones conservan la forma local de áreas pequeñas. Estas proyecciones conservan ángulos individuales para describir relaciones espaciales mostrando líneas perpendiculares de red geográfica que forman intersección en ángulos de 90 grados en el mapa. Se conservan todos los ángulos; sin embargo, el área del mapa está distorsionada. Las proyecciones *Mercator* y *Cónica conforme de Lambert* son ejemplos de proyecciones conformes.

Proyecciones equidistantes
Estas proyecciones conservan las distancias entre ciertos puntos manteniendo la escala de un conjunto de datos determinado. Algunas de las distancias serán distancias verdaderas, que son las mismas distancias en la misma escala que el globo. Si sale fuera del conjunto de datos, la escala se distorsionará más. La proyección *sinusoidal* y la proyección *cónica equidistante* son ejemplos de proyecciones equidistantes.

Proyecciones de dirección verdadera y azimutales
Estas proyecciones conservan la dirección de un punto a otros puntos manteniendo algunos de los arcos de círculo grandes. Estas proyecciones dan correctamente las direcciones o azimuts de todos los puntos del mapa respecto al centro. Los mapas azimutales se pueden combinar con proyecciones de áreas iguales, conformes y equidistantes. La proyección *azimutal de igual área de Lambert* y la proyección *azimutal equidistante* son ejemplos de proyecciones azimutales.

Conceptos relacionados:

- "Sistema de coordenadas geográficas" en la página 60
- "Sistemas de coordenadas" en la página 59

Tareas relacionadas:

- "Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 141

Información relacionada:

- "Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos" en la página 131
- "ST_create_coordsys" en la página 244

• "Sistemas de coordenadas soportados" en la página 539

Selección o creación de sistemas de coordenadas

Después de habilitar una base de datos para operaciones espaciales, está listo para planear proyectos que utilicen datos espaciales. Un primer paso en la planificación de un proyecto es determinar qué sistema de coordenadas se debe utilizar. Las opciones son las siguientes:

- Puede utilizar un sistema de coordenadas proporcionado con DB2 Spatial Extender o uno que haya sido creado por un usuario. Con DB2 Spatial Extender se proporcionan más de 2000 sistemas de coordenadas. Entre éstos se encuentran:
  - Un sistema de coordenadas al que DB2 Spatial Extender se refiere como "Sin especificar". Utilice este sistema de coordenadas cuando:
    - Necesite definir ubicaciones que no tengan una relación directa con la superficie de la tierra; por ejemplo, ubicaciones de oficinas situadas dentro de un edificio de oficinas o ubicaciones de estanterías situadas dentro de un almacén.
    - Puede definir estas ubicaciones en términos de coordenadas positivas que incluyan pocos valores decimales o ninguno.
  - GCS_NORTH_AMERICAN_1983. Utilice este sistema de coordenadas cuando necesite definir ubicaciones en los Estados Unidos; por ejemplo:
    - Cuando importe datos espaciales para los Estados Unidos desde los CD de "Mapas y datos" que se proporcionan con DB2 Spatial Extender.
    - Cuando planee utilizar el geocodificador proporcionado con DB2 Spatial Extender para geocodificar direcciones de los Estados Unidos.

  Para saber más sobre los sistemas de coordenadas, y para determinar qué otros sistemas de coordenadas se proporcionan con DB2 Spatial Extender, y qué otros sistemas de coordenadas han sido creados por otros usuarios (si se ha creado alguno), consulte la vista de catálogo DB2SE.ST_COORDINATE_SYSTEMS.
- Puede crear un sistema de coordenadas.

Requisitos previos:

Antes de crear un sistema de coordenadas, el ID de usuario debe tener autorización SYSADM o DBADM sobre la base de datos que se ha habilitado para operaciones espaciales. No se necesita ninguna autorización para utilizar un sistema de coordenadas existente.

Procedimiento:

Puede crear un sistema de coordenadas de cualquiera de las maneras siguientes:

- Créelo desde la ventana Crear sistema de coordenadas del Centro de control de DB2.
- Emita el mandato **db2se create_cs** desde el procesador de línea de mandatos db2se.
- Ejecute una aplicación que invoque al procedimiento almacenado db2se.ST_create_coordsys.

Conceptos relacionados:

• "Sistemas de coordenadas" en la página 59

Tareas relacionadas:

- "Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 141

Información relacionada:

- "Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos" en la página 131
- "ST_create_coordsys" en la página 244

Cómo configurar sistemas de referencia espacial

Cuando planea un proyecto que utiliza datos espaciales, es necesario determinar si alguno de los sistemas de referencia espacial disponible para el usuario se puede utilizar para estos datos. Si ninguno de los sistemas disponibles es adecuado para los datos, puede crear uno que lo sea. Esta sección explica el concepto de sistemas de referencia espacial y describe las tareas de seleccionar cuál utilizar y de crear uno.

Sistemas de referencia espacial

Un *sistema de referencia espacial* es un conjunto de parámetros que incluye:

- El nombre del sistema de coordenadas del cual se obtienen las coordenadas.
- El identificador numérico que identifica de forma exclusiva al sistema de referencia espacial.
- Coordenadas que definen la máxima extensión de espacio posible a la que se hace referencia mediante un rango determinado de coordenadas.
- Números que, cuando se utilizan en ciertas operaciones matemáticas, convierten las coordenadas recibidas como datos de entrada en valores que se pueden procesar con máxima eficacia.

Las siguientes secciones abordan los valores de parámetros que definen un identificador, una extensión de espacio máxima y factores de conversión.

Indentificador de sistemas de referencia espacial:

El identificador de sistemas de referencia espacial (SRID) se utiliza como parámetro de entrada para varias funciones espaciales.

Para un sistema de referencia espacial, el valor del SRID debe estar comprendido entre 2000000000 y 2000001000. DB2® Geodetic Extender proporciona 318 sistemas de referencia espacial (SRS) geodésicos. Para obtener más información, consulte el apartado "DB2 Geodetic Extender" en la página 165.

Definición del espacio que comprende coordenadas almacenadas en una columna espacial:

Las coordenadas de una columna espacial definen normalmente ubicaciones que extienden a lo largo de la tierra. El espacio cubierto por la extensión —de este a oeste y de norte a sur— se denomina *extensión espacial*. Por ejemplo, imagine una planicie aluvial cuyas coordenadas se almacenan en una columna espacial. Imagine que las coordenadas situadas más al oeste y más al este son los valores de latitud de –24,556 y –19,338, respectivamente, y que las coordenadas situadas más al norte y más al sur son los valores de longitud de 18,819 y 15,809 grados, respectivamente. La extensión espacial de la planicie aluvial es un espacio que se extiende en un plano oeste-este entre las dos latitudes y en un plano norte-sur

entre las dos longitudes. Puede incluir estos valores en un sistema de referencia espacial asignándolos a ciertos parámetros. Si la columna espacial incluye medidas y coordenadas Z, necesitará incluir también las coordenadas y las medidas Z más altas y más bajas en el sistema de referencia espacial.

El término *extensión espacial* se puede referir no solamente a una extensión real de ubicaciones, como en el párrafo anterior, sino también a una extensión potencial. Imagine que se espera que las planicies aluviales del ejemplo anterior se ensanchen en los próximos cinco años. Podría calcular cuáles serían al final del quinto año las coordenadas situadas más al oeste, este, norte y sur de las planicies. A continuación, podría asignar estos cálculos, en lugar de las coordenadas actuales, a los parámetros para una extensión espacial. De esta manera, podría conservar el sistema de referencia espacial conforme las planicies se ensanchen y sus latitudes y longitudes mayores se añadan a la columna espacial. En caso contrario, si el sistema de referencia espacial estuviera limitado a las latitudes y longitudes originales, sería necesario modificarlo o sustituirlo conforme las planicies aluviales crecieran.

Conversión a valores que mejoran el rendimiento:

Normalmente, la mayoría de coordenadas en un sistema de coordenadas son valores decimales; algunos son números enteros. Además, las coordenadas situadas al este del origen son valores positivos; aquellas situadas al oeste son valores negativos. Antes de ser almacenadas por Spatial Extender, las coordenadas con valores negativos se convierten a valores positivos y las coordenadas con valores decimales se convierten en números enteros. Como resultado, Spatial Extender almacena todas las coordenadas como valores enteros positivos. El propósito de esta conversión es mejorar el rendimiento al procesar las coordenadas.

Algunos parámetros de un sistema de referencia espacial se utilizan para realizar las conversiones descritas en el párrafo anterior. Un parámetro, denominado *desplazamiento*, se resta de cada coordenada negativa, dejando un valor positivo como resultado. Cada coordenada decimal se multiplica por otro parámetro, denominado *factor de escala*, dando como resultado un número entero cuya precisión es la misma que la de la coordenada decimal. (El desplazamiento se resta de las coordenadas positivas tanto como de las negativas; las coordenadas no decimales, así como las coordenadas decimales, se multiplican por el factor de escala. De esta forma, todas las coordenadas positivas y no decimales permanecen en proporción con las coordenadas negativas y decimales.)

Estas conversiones se realizan internamente y permanecen vigentes sólo hasta que se recuperan las coordenadas. Los datos de entrada y los resultados de las consultas siempre contienen coordenadas en su forma original y no convertida.

Conceptos relacionados:

- “Factores de conversión que transforman datos de coordenadas en números enteros” en la página 74
- “Sistemas de coordenadas” en la página 59

Tareas relacionadas:

- “Cómo decidirse entre la utilización de un sistema de referencia espacial por omisión o la creación de un sistema nuevo” en la página 70
- “Creación de un sistema de referencia espacial” en la página 76

Cómo decidirse entre la utilización de un sistema de referencia espacial por omisión o la creación de un sistema nuevo

Después de determinar el sistema de coordenadas que utilizará, ya está preparado para proporcionar un sistema de referencia espacial adecuado para los datos de coordenadas con los que está trabajando. DB2 Spatial Extender proporciona cinco sistemas de referencia espacial para datos espaciales y DB2 Geodetic Extender proporciona 318 sistemas de referencia espacial geodésica para datos geodésicos.

Procedimiento:

Para determinar si puede utilizar uno de los sistemas de referencia espacial por omisión o uno de los sistemas de referencia geodésicos predefinidos:

1. Conteste a las siguientes preguntas:
   - ¿El sistema de coordenadas en el que está basado el sistema de referencia espacial por omisión cubre la zona geográfica con la que está trabajando?

     Estos sistemas de coordenadas se muestran en el apartado "Sistemas de referencia espacial suministrados con DB2 Spatial Extender" en la página 71.
   - ¿Se encuentran sus datos en un sistema de coordenadas geográficas que utiliza grados decimales o grads como unidades de medida? ¿Los datos abarcan una porción grande de la superficie de la tierra? ¿Necesita realizar cálculos precisos sobre distancia, longitud y área? ¿Alguno de los datos se encuentra cerca del polo norte, del polo sur o de la línea de datos internacional?

     Si ha contestado afirmativamente a alguna de estas preguntas, probablemente deseará utilizar uno de los 318 sistemas de referencia espacial geodésicos. Para obtener información sobre dichos sistemas de referencia espacial geodésicos, consulte el apartado "Sistemas de referencia soportados por DB2 Geodetic Extender" en la página 220.
   - ¿Los factores de conversión asociados con uno de los sistemas de referencia espacial por omisión funcionan con sus datos de coordenadas?

     Spatial Extender utiliza valores de *desplazamiento* y factores de *escala* para convertir los datos de coordenadas proporcionados en números enteros positivos. Para determinar si sus datos de coordenadas funcionan con los valores de desplazamiento y los factores de escala proporcionados para uno de los sistemas de referencia espacial por omisión:

     a. Repase la información del apartado "Factores de conversión que transforman datos de coordenadas en números enteros" en la página 74.
     b. Observe cómo están definidos estos factores para los sistemas de referencia espacial por omisión. Si, después de aplicar el valor de desplazamiento a las coordenadas X e Y mínimas, ninguna de estas dos coordenadas es superior a 0, deberá crear un sistema de referencia espacial nuevo y definir los desplazamientos por sí mismo. Para obtener más información sobre cómo crear un nuevo sistema de referencia espacial, consulte el apartado "Creación de un sistema de referencia espacial" en la página 76.
   - ¿Los datos con los que está trabajando incluyen coordenadas de altura y profundidad (coordenadas Z) o medidas (coordenadas M)?

     Si trabaja con coordenadas Z o M, es posible que deba crear un nuevo sistema de referencia espacial factores de escala y desplazamientos Z o M adecuados para sus datos.

2. Si los sistemas de referencia espacial o los sistemas de referencia geodésicos existentes no funcionan con sus datos, deberá "Creación de un sistema de referencia espacial" en la página 76.

Después de decidir qué sistema de referencia espacial necesita, deberá especificar su elección a Spatial Extender cuando realice una de las siguientes tareas:

- "Creación de columnas espaciales" en la página 86
- "Registro de columnas espaciales" en la página 87

Conceptos relacionados:

- "Factores de conversión que transforman datos de coordenadas en números enteros" en la página 74
- "Cuándo utilizar DB2 Geodetic Extender y DB2 Spatial Extender" en la página 166
- "Sistemas de referencia espacial" en la página 68

Tareas relacionadas:

- "Creación de un sistema de referencia espacial" en la página 76
- "Creación de columnas espaciales" en la página 86
- "Registro de columnas espaciales" en la página 87
- "Creación de un sistema de referencia espacial"
- "Registro de una columna espacial con un sistema de referencia espacial"
- "Selección de un sistema de referencia espacial"

Información relacionada:

- "Sistemas de referencia espacial suministrados con DB2 Spatial Extender" en la página 71
- "Sistemas de referencia soportados por DB2 Geodetic Extender" en la página 220

Sistemas de referencia espacial suministrados con DB2 Spatial Extender

DB2 Spatial Extender proporciona los sistemas de referencia espacial que se muestran en la tabla inferior, junto con el sistema de coordenadas en el que se basa cada sistema de referencia espacial y los valores de desplazamiento y los factores de escala que DB2 Spatial Extender utiliza para convertir los datos de coordenadas en números enteros positivos. Puede encontrar información sobre estos sistemas de referencia espacial en la vista de catálogos DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Si trabaja con grados decimales (todos los datos de los CD de datos de ejemplo de DB2 Spatial Extender están en grados decimales), los valore de desplazamiento y los factores de escala de los sistemas de referencia espacial por omisión darán soporte a una amplia gama de coordenadas de latitud-longitud y conservarán 6 posiciones decimales, equivalente a 10 cm aproximadamente.

Si piensa utilizar el geocodificador, que sólo funciona con direcciones de los EE.UU., asegúrese de seleccionar o crear un sistema de referencia espacial que maneje coordenadas de los EE.UU., como por ejemplo el sistema de coordenadas GCS_NORTH_AMERICAN_1983. Si no especifica de qué sistema de coordenadas deberían derivarse los datos espaciales, Spatial Extender utiliza el sistema de referencia espacial DEFAULT_SRS.

Utilice la tabla que aparece a continuación para decidir si va a utilizar un sistema de referencia espacial por omisión o si va a crear un nuevo sistema (consulte el apartado "Cómo decidirse entre la utilización de un sistema de referencia espacial por omisión o la creación de un sistema nuevo" en la página 70. Si ninguno de los sistemas de referencia espacial por omisión se ajusta a sus necesidades, puede crear un sistema de referencia espacial nuevo. Consulte el apartado "Creación de un sistema de referencia espacial" en la página 76 para obtener más información.

Tabla 4. Sistemas de referencia espacial proporcionados con DB2 Spatial Extender

Sistema de referencia espacial	ID de SRS	Sistema de coordenadas	Valores de desplaza-miento	Factores de escala	Cuándo se utiliza
DEFAULT _SRS	0	Ninguno	xOffset = 0 yOffset = 0 zOffset = 0 mOffset = 0	xScale = 1 yScale = 1 zScale = 1 mScale = 1	Puede seleccionar este sistema cuando sus datos sean independientes de un sistema de coordenadas o no pueda o no necesite especificar uno.
NAD83_ SRS_1	1	GCS_NORTH _AMERICAN _1983	xOffset = –180 yOffset = –90 zOffset = 0 mOffset = 0	xScale = 1,000,000 yScale = 1,000,000 zScale = 1 mScale = 1	Puede seleccionar este sistema de referencia espacial si va a utilizar los datos de ejemplo de los EE.UU. suministrados con DB2 Spatial Extender. Si los datos de coordenadas con los que está trabajando se recopilaron después de 1983, utilice este sistema en lugar de NAD27_SRS_1002.

Tabla 4. Sistemas de referencia espacial proporcionados con DB2 Spatial Extender (continuación)

Sistema de referencia espacial	ID de SRS	Sistema de coordenadas	Valores de desplaza-miento	Factores de escala	Cuándo se utiliza
NAD27_ SRS_1002	1002	GCS_NORTH _AMERICAN _1927	xOffset = –180 yOffset = –90 zOffset = 0 mOffset = 0	xScale = 5,965,232 yScale = 5,965,232 zScale = 1 mScale = 1	Puede seleccionar este sistema de referencia espacial si va a utilizar los datos de ejemplo de los EE.UU. suministrados con DB2 Spatial Extender. Si los datos de coordenadas con los que está trabajando se recopilaron antes de 1983, utilice este sistema en lugar de NAD83_SRS_1. Este sistema proporciona un mayor grado de precisión que el resto de sistemas de referencia espacial por omisión.
WGS84_ SRS_1003	1003	GCS_WGS _1984	xOffset = –180 yOffset = –90 zOffset = 0 mOffset = 0	xScale = 5,965,232 yScale = 5,965,232 zScale = 1 mScale = 1	Puede seleccionar este sistema de referencia espacial si trabaja con datos que no sean de los EE.UU. (Este sistema maneja coordenadas de todo el mundo). No utilice este sistema si va a utilizar el geocodificador por omisión suministrado con DB2 Spatial Extender, porque el geocodificador sólo sirve para direcciones de los EE.UU.

Tabla 4. Sistemas de referencia espacial proporcionados con DB2 Spatial Extender (continuación)

Sistema de referencia espacial	ID de SRS	Sistema de coordenadas	Valores de desplaza-miento	Factores de escala	Cuándo se utiliza
DE_HDN _SRS_1004	1004	GCSW _DEUTSCHE _HAUPTDRE IECKSNETZ	xOffset = –180 yOffset = –90 zOffset = 0 mOffset = 0	xScale = 5,965,232 yScale = 5,965,232 zScale = 1 mScale = 1	Este sistema de referencia espacial está basado en un sistema de coordenadas para direcciones alemanas.

Conceptos relacionados:

- "Sistemas de referencia espacial" en la página 68

Información relacionada:

- "Vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS" en la página 303

Factores de conversión que transforman datos de coordenadas en números enteros

DB2® Spatial Extender utiliza valores de *desplazamiento* y factores de *escala* para convertir en números enteros positivos los datos de coordenadas proporcionados. Los sistemas de referencia espacial por omisión ya tienen valores de desplazamiento y factores de escala asociados a los mismos. Si está creando un nuevo sistema de referencia espacial, deberá determinar los factores de escala y, opcionalmente, los factores de desplazamiento que funcionen mejor con sus datos. Para obtener más información, consulte el apartado "Creación de un sistema de referencia espacial" en la página 76.

Valores de desplazamiento

Un valor de desplazamiento es un número que se resta de todas las coordenadas, dejando sólo valores positivos como resto. Spatial Extender convierte los datos de coordenadas utilizando las siguientes fórmulas para garantizar que todos los valores de coordenadas ajustados sean mayores que 0.

Notación de la fórmula: En estas fórmulas, la notación "mín" representa "el mínimo de todos". Por ejemplo, "mín(x)" significa "el mínimo de todas las coordenadas x". El desplazamiento de cada dirección geográfica se representa como *dimensión*Offset. Por ejemplo, xOffset es el valor de desplazamiento aplicado a todas las coordenadas X.

```
mín(x) – xOffset ≥ 0
mín(y) – yOffset ≥ 0
mín(z) – zOffset ≥ 0
mín(m) – mOffset ≥ 0
```

Factores de escala

Un factor de escala es un valor que, multiplicado por medidas y coordenadas decimales, da lugar a números enteros que tienen como mínimo el mismo número de dígitos significantes que las medidas y coordenadas originales. Spatial Extender convierte las coordenadas decimales utilizando las siguientes fórmulas para

garantizar que todos los valores de coordenadas ajustados sean números enteros positivos. Los valores convertidos no pueden sobrepasar 2^{53} (aproximadamente 9 * 10^{15}).

Notación de la fórmula: En estas fórmulas la notación "máx" representa "el máximo de todos". El desplazamiento para cada dimensión geográfica se representa como *dimensión*Offset (por ejemplo, xOffset es el valor de desplazamiento aplicado a todas las coordenadas X). El factor de escala para cada dimensión geográfica se representa como *dimensión*Scale (por ejemplo, xScale es el factor de escala aplicado a todas las coordenadas X).

$(máx(x) - xOffset) * xScale \leq 2^{53}$
$(máx(y) - yOffset) * yScale \leq 2^{53}$
$(máx(z) - zOffset) * zScale \leq 2^{53}$
$(máx(m) - mOffset) * mScale \leq 2^{53}$

Cuando seleccione qué factores de escala funcionan mejor con sus datos de coordenadas, asegúrese de:

- Utilizar el mismo factor de escala para las coordenadas X e Y.
- Que cuando se multiplique por una coordenada decimal X o por una coordenada decimal Y, el factor de escala dé como resultado un valor menor que 2^{53}. Una técnica habitual consiste en convertir el factor de escala en una potencia de 10. Es decir, el factor de escala deberá ser 10 a la primera potencia (10), 10 a la segunda potencia (100), 10 a la tercera potencia (1000) o, si es necesario, un factor superior.
- Que el factor de escala sea lo suficientemente grande como para garantizar que el número de dígitos del nuevo número entero sea el mismo que el de dígitos significantes de la coordenada decimal original.

Ejemplo:

Supongamos que la entrada de la función ST_Point consta de una coordenada X de 10.01 y una coordenada Y de 20.03, además del identificador de un sistema de referencia espacial. Cuando se invoca a ST_Point, la función multiplica el valor 10.01 y el valor 20.03 por el factor de escala del sistema de referencia espacial para las coordenadas X e Y. Si este factor de escala es 10, los números enteros resultantes que almacenará Spatial Extender serán 100 y 200, respectivamente. Debido a que el número de dígitos significantes de estos enteros (3) es menor que el número de dígitos significantes de las coordenadas (4), Spatial Extender no podrá revertir estos enteros a las coordenadas originales ni obtener a partir de ellos valores que sean coherentes con el sistema de coordenadas al que pertenecen dichas coordenadas. Sin embargo, si el factor de escala es 100, los números enteros resultantes que almacenará DB2 Spatial Extender serán 1001 y 2003, valores que se pueden revertir a las coordenadas originales o de los que se pueden obtener coordenadas compatibles.

Unidades para los valores de desplazamiento y los factores de escala

Tanto si utiliza un sistema de referencia espacial existente como si crea uno nuevo, las unidades para los valores de desplazamiento y los factores de escala variarán según el tipo de sistema de coordenadas que utilice. Por ejemplo, si utiliza un sistema de coordenadas geográficas, los valores estarán en unidades angulares, como son los grados decimales; si utiliza un sistema de coordenadas proyectadas, los valores estarán en unidades lineales, como son los metros o los pies.

Tareas relacionadas:

- "Cómo decidirse entre la utilización de un sistema de referencia espacial por omisión o la creación de un sistema nuevo" en la página 70

Creación de un sistema de referencia espacial

Si ninguno de los sistemas de referencia espacial proporcionados con DB2 Spatial Extender funciona con sus datos, deberá crear un nuevo sistema de referencia espacial.

Procedimiento:

Para crear un sistema de referencia espacial nuevo:

1. Elija la interfaz.

   Puede crear un sistema de referencia espacial de cualquiera de las maneras siguientes:

   - Utilice la ventana Crear sistema de referencia espacial del Centro de control de DB2. Para obtener más información sobre cómo utilizar esta ventana, consulte la ayuda en línea.
   - Emita el mandato **db2se create_srs** desde el procesador de línea de mandatos db2se. Para obtener más información, consulte el apartado "Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos" en la página 131.
   - Ejecute una aplicación que invoque al procedimiento almacenado db2se.ST_create_srs. Para obtener más información, consulte el apartado "ST_create_srs" en la página 246.

2. Especifique un ID de sistema de referencia espacial (SRID) apropiado:

   - Para los datos geodésicos en una representación terrestre redonda, especifique un valor de SRID comprendido entre 200000318 y 2000001000.
   - Para los datos espaciales en una representación terrestre plana, especifique un SRID que no esté todavía definido.

3. Decida el grado de precisión que desea. Tiene estas dos opciones:

   - Especificar la extensión de la zona geográfica con la que esté trabajando y los factores de escala que desee utilizar con los datos de coordenadas. Spatial Extender toma las extensiones que ha especificado y calcula el desplazamiento.

     Puede especificar extensiones de una de las maneras siguientes:

     – Elija **Extensiones** en la ventana Crear sistema de referencia espacial del Centro de control.

     – Proporcione los parámetros apropiados para el mandato **db2se create_srs** o el procedimiento almacenado db2se.ST_create_srs.

   - Especificar tanto los valores de desplazamiento (necesarios para que Spatial Extender convierta valores negativos en valores positivos) como los factores de escala (necesarios para que Spatial Extender convierta valores decimales en números enteros). Utilice este método cuando necesite seguir criterios estrictos para lograr mayor precisión.

     Puede especificar valores de desplazamiento y factores de escala de una de las maneras siguientes:

     – Elija **Desplazamiento** en la ventana Crear sistema de referencia espacial del Centro de control

     – Proporcione los parámetros apropiados para el **mandato db2se create_srs** o el procedimiento almacenado db2se.ST_create_srs.

Para obtener más información, consulte el apartado "Factores de conversión que transforman datos de coordenadas en números enteros" en la página 74.

4. Calcule la información de conversión que Spatial Extender necesita para convertir datos de coordenadas en números enteros positivos y proporcione esta información mediante la interfaz de su elección. Esta información será distinta según el método que elija en el paso 3.
   - Si elige el método "Extensiones" en el paso 3, deberá calcular la información siguiente:
     - Factores de escala. Si cualquiera de las coordenadas con las que trabaja está en valores decimales, calcule los factores de escala (consulte el apartado "Cálculo de los factores de escala" en la página 78. Los factores de escala son números que, multiplicados por medidas y coordenadas decimales, dan lugar a números enteros que tienen como mínimo el mismo número de dígitos significantes que las medidas y coordenadas originales. Si las coordenadas son números enteros, los factores de escala se pueden establecer en 1. Si las coordenadas son valores decimales, el factor de escala se debe establecer en un número que convierta la parte decimal en un valor entero. Por ejemplo, si las unidades de coordenadas son metros y la exactitud de los datos es 1 cm., necesitará un factor de escala de 100.
     - Valores mínimos y máximos para las coordenadas y las medidas. (Consulte el apartado "Determinación de las coordenadas y medidas mínimas y máximas" en la página 79 para obtener más información).
   - Si elige el método "Desplazamiento " en el paso 3, deberá calcular la información siguiente:
     - Valores de desplazamiento

       Si los datos de coordenadas incluyen números o medidas negativos, deberá especificar los valores de desplazamiento que desee utilizar. Un desplazamiento es un número que se resta de todas las coordenadas, dejando sólo valores positivos como resto. Si trabaja con coordenadas positivas, establezca todos los valores de desplazamiento en 0. Si no trabaja con coordenadas positivas, seleccione un desplazamiento que, aplicado a los datos de coordenadas, dé como resultado números enteros que sean menores que el valor del mayor número entero positivo (9,007,199,254,740,992). (Consulte el apartado "Cálculo de valores de desplazamiento" en la página 80 para obtener más información).
     - Factores de escala

       Si alguna de las coordenadas para las ubicaciones que está representando es un número decimal, determine qué factores de escala se deben utilizar y entre estos factores de escala en la ventana Crear sistema de referencia espacial. Consulte el apartado "Cálculo de los factores de escala" en la página 78.
5. Emita el mandato **db2se create_srs** o el procedimiento almacenado db2se.ST_create_srs.

   Por ejemplo, el siguiente mandato crear un sistema de referencia espacial denominado misrs:

   ```
   db2se create_srs mi_bd -srsName \"misrs\"
    -srsID 100 -xScale 10 -coordsysName
   \"GCS_North_American_1983\"
   ```

   Para obtener más información sobre el modo de ejecutar una aplicación que invoque al procedimiento almacenado db2se.ST_create_srs, consulte "ST_create_srs" en la página 246.

Después de crear un sistema de referencia espacial, debe asociarlo con una columna espacial utilizando una de las siguientes tareas:

- "Creación de columnas espaciales" en la página 86
- "Registro de columnas espaciales" en la página 87

Conceptos relacionados:

- "Factores de conversión que transforman datos de coordenadas en números enteros" en la página 74
- "Sistemas de coordenadas" en la página 59
- "Sistemas de referencia espacial" en la página 68

Tareas relacionadas:

- "Cálculo de los factores de escala" en la página 78
- "Determinación de las coordenadas y medidas mínimas y máximas" en la página 79
- "Cálculo de valores de desplazamiento" en la página 80
- "Creación de columnas espaciales" en la página 86
- "Registro de columnas espaciales" en la página 87
- "Creación de un sistema de referencia espacial"
- "Registro de una columna espacial con un sistema de referencia espacial"

Información relacionada:

- "Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos" en la página 131
- "ST_create_srs" en la página 246

Cálculo de los factores de escala

Si crea un sistema de referencia espacial y cualquiera de las coordenadas con las que está trabajando son valores decimales, calcule los factores de escala apropiados para las coordenadas y las medidas utilizadas. Los factores de escala son números que, multiplicados por medidas y coordenadas decimales, dan lugar a números enteros que tienen como mínimo el mismo número de dígitos significantes que las medidas y coordenadas originales.

Requisitos previos:

Antes de calcular los factores de escala que funcionarán con sus datos, asegúrese de conocer las directrices para la elección de los "Factores de conversión que transforman datos de coordenadas en números enteros" en la página 74.

Procedimiento:

Para calcular los factores 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. Busque la coordenada decimal que tenga la precisión decimal más larga. A continuación, determine por qué potencia de 10 se puede multiplicar esta coordenada para obtener un número entero de igual precisión. Por ejemplo, de

las tres coordenadas decimales del ejemplo actual, 5,1235 tiene la precisión decimal más larga. Si la multiplicamos por 10 a la cuarta potencia (10000), obtendremos el número entero 51235.

3. Determine qué entero generado por la multiplicación que se acaba de describir es menor que 2^{53}. 51235 no es demasiado grande. Pero supongamos que, además de 1,23, 5,11235 y 6,789, su rango de coordenadas X e Y incluye un cuarto valor decimal, 10000000006,789876. Puesto que la precisión decimal de esta coordenada es mayor que la de las otras tres, tendrá que multiplicar *esta* coordenada—nt 5.1235—por una potencia de 10. Para convertirla en un número entero, la tendría que multiplicar por 10 a la sexta potencia (1000000). Sin embargo, el valor resultante, 10000000006789876, es mayor que 2^{53}. Si DB2 Spatial Extender intentara almacenarlo, los resultados serían imprevisibles.

   Para evitar este problema, seleccione una potencia de 10 que, multiplicada por la coordenada original, dé como resultado un número decimal que DB2 Spatial Extender pueda truncar en un número entero susceptible de almacenamiento, con una pérdida de precisión mínima. En este caso, podría seleccionar 10 a la quinta potencia (100000). Multiplicando 100000 por 10000000006.789876 obtendrá 1000000000678987.6. DB2 Spatial Extender redondearía este número a 1000000000678988, reduciendo ligeramente su precisión.

Después de calcular factores de escala, deberá determinar los valores de extensión (consulte el apartado "Determinación de las coordenadas y medidas mínimas y máximas". A continuación, emita el mandato **db2se create_srs** o el procedimiento almacenado db2se.ST_create_srs.

Conceptos relacionados:

- "Factores de conversión que transforman datos de coordenadas en números enteros" en la página 74
- "Sistemas de referencia espacial" en la página 68

Tareas relacionadas:

- "Determinación de las coordenadas y medidas mínimas y máximas" en la página 79

Determinación de las coordenadas y medidas mínimas y máximas

Determine las coordenadas y medidas máximas y mínimas si decide especificar transformaciones de extensión cuando cree un sistema de referencia espacial.

Requisitos previos:

Utilice este procedimiento para determinar coordenadas y medidas mínimas y máximas si:

- Decide crear un nuevo sistema de referencia espacial porque ninguno de los sistemas de referencia espacial proporcionado con DB2 Spatial Extender funcionará con sus datos. Para obtener más información, consulte el apartado "Cómo decidirse entre la utilización de un sistema de referencia espacial por omisión o la creación de un sistema nuevo" en la página 70.
- Decide utilizar transformaciones de extensión para convertir las coordenadas.

Procedimiento:

Para determinar las coordenadas y medidas mínimas y máximas de las ubicaciones que desee representar:

- Determine las coordenadas X mínimas y máximas.

  Para hallar la coordenada X mínima, identifique la coordenada X de su dominio que se encuentre más al oeste. (Si la ubicación se encuentra al oeste del punto de origen, esta coordenada será un valor negativo). Para hallar la coordenada X máxima, identifique la coordenada X de su dominio que se encuentre más al este. Por ejemplo, si está representando pozos de petróleo y cada uno de ellos está definido por un par de coordenadas X e Y, la coordenada X que indica la ubicación del pozo de petróleo que está situado más al oeste es la coordenada X mínima y la coordenada X que indica la ubicación del pozo de petróleo situado más al este es la coordenada X máxima.
- Determine las coordenadas mínimas y máximas.

  Para localizar la coordenada Y mínima, identifique la coordenada Y de su dominio que se encuentre más al sur. (Si la ubicación se encuentra al sur del punto de origen, esta coordenada será un valor negativo). Para determinar la coordenada Y máxima, identifique la coordenada Y de su dominio que se encuentre más al sur.
- Determine las coordenadas Z mínimas y máximas.

  La coordenada Z mínima es la coordenada de profundidad mayor y la coordenada Z máxima es la coordenada de altura mayor.
- Determine las medidas mínimas y máximas

  Si va a incluir medidas en los datos espaciales, determine qué medida tiene el valor numérico más alto y qué medida tiene el más bajo.

Para los tipos formados por varios elementos, como los multipolígonos, asegúrese de seleccionar el punto más lejano del polígono más lejano en la dirección que esté calculando. Por ejemplo, si intenta identificar la coordenada X mínima, identifique la coordenada X que se encuentre más al oeste del polígono que esté más al oeste en el multipolígono.

Una vez determinados los valores de extensión, si alguna de las coordenadas es un valor decimal, deberá calcular factores de escala (consulte el apartado "Cálculo de los factores de escala" en la página 78. De lo contrario, emita el mandato **db2se create_srs** o el procedimiento almacenado db2se.ST_create_srs.

Tareas relacionadas:

- "Cómo decidirse entre la utilización de un sistema de referencia espacial por omisión o la creación de un sistema nuevo" en la página 70
- "Cálculo de los factores de escala" en la página 78

Cálculo de valores de desplazamiento

Si crea un sistema de referencia espacial y los datos de coordenadas incluyen números negativos, deberá especificar los valores de desplazamiento que desee utilizar. Un desplazamiento es un número que se resta de todas las coordenadas, dejando sólo valores positivos como resto. Puede mejorar el rendimiento de las operaciones espaciales si las coordenadas son enteros positivos en vez de números o medidas negativas.

Requisitos previos:

Si los datos de coordenadas incluyen números o medidas negativos, deberá especificar valores de desplazamiento.

Procedimiento:

Para calcular los valores de desplazamiento para las coordenadas con las que esté trabajando:

1. Determine las coordenadas negativas X, Y y Z más bajas dentro del rango de coordenadas para las ubicaciones que desea representar. Si los datos van a incluir medidas negativas, determine las más bajas de estas medidas. Consulte el apartado "Determinación de las coordenadas y medidas mínimas y máximas" en la página 79.
2. Opcional pero recomendado: Indique a DB2 Spatial Extender que el dominio que abarca las ubicaciones que le interesan es mayor de lo que es en realidad. En consecuencia, después de escribir datos acerca de estas ubicaciones en la columna espacial, puede añadir datos acerca de ubicaciones de nuevos elementos conforme éstos se añaden a los bordes más exteriores del dominio, sin necesidad de sustituir el sistema de referencia espacial por otro.

   Para cada coordenada y medida que haya identificado en el paso 1, sume una cantidad comprendida entre el cinco y el diez por ciento de la coordenada o medida. El resultado se conoce como un *valor aumentado*. Por ejemplo, si la coordenada X negativa más baja es –100, puede sumar –5 a la misma, obteniendo como resultado un valor aumentado de –105.Posteriormente, cuando cree un sistema de referencia espacial, indicará que la coordenada X más baja es –105, en lugar del valor real de –100.DB2 Spatial Extender interpretará entonces –105 como el límite situado más al oeste del dominio.
3. Encuentre un valor que, al restarlo al valor aumentado de X, dé un resultado de cero; este será el valor de desplazamiento para las coordenadas X. DB2 Spatial Extender resta este número a todas las coordenadas X para producir sólo valores positivos.

   Por ejemplo, si el valor aumentado de X es –105, tiene que restarle –105 para obtener 0. DB2 Spatial Extender restará entonces –105 de todas las coordenadas X asociadas con los elementos representados. Puesto que ninguna de estas coordenadas es mayor que –100, todos los valores resultantes de la resta serán positivos.
4. Repita el paso 3 para el valor Y aumentado, el valor Z aumentado y la medida aumentada.

Después de calcular los valores de desplazamiento, cree un sistema de referencia espacial (consulte el apartado "Creación de un sistema de referencia espacial" en la página 76).

Tareas relacionadas:

- "Determinación de las coordenadas y medidas mínimas y máximas" en la página 79
- "Creación de un sistema de referencia espacial" en la página 76

Capítulo 9. Configuración de columnas espaciales

En la preparación para obtener datos espaciales para un proyecto, no sólo selecciona o crea un sistema de coordenadas y un sistema de referencia espacial; también proporciona una o varias columnas de tabla donde contener los datos. Este capítulo:

- Indica que los resultados de las consultas de las columnas se pueden obtener gráficamente y proporciona directrices para seleccionar tipos de datos para las columnas
- Describe la tarea de proporcionar las columnas
- Describe la tarea de hacer las columnas accesibles a herramientas que puedan visualizar su contenido en forma gráfica

Columnas espaciales

Columnas espaciales con contenido visualizable

Cuando utiliza una herramienta de visualización, como por ejemplo ArcExplorer for DB2®, para consultar una columna espacial, la herramienta devolverá resultados en forma de una representación gráfica; por ejemplo, un mapa de límites de parcelas o el esquema de un sistema de carreteras. Algunas herramientas de visualización requieren que todas las filas de la columna utilicen el mismo sistema de referencia espacial. Para aplicar esta restricción, debe registrar la columna en un sistema de referencia espacial.

Tipos de datos espaciales

Cuando habilita una base de datos para operaciones espaciales, DB2 Spatial Extender proporciona a la base de datos una jerarquía de tipos de datos estructurados. La Figura 12 en la página 84 presenta esta jerarquía. En esta figura, los tipos de los que pueden crearse instancias tienen un fondo blanco; los tipos de los que no pueden crearse instancias tienen un fondo gris.

Los tipos de datos de los que pueden crearse instancias son ST_Point, ST_LineString, ST_Polygon, ST_GeomCollection, ST_MultiPoint, ST_MultiPolygon y ST_MultiLineString.

Los tipos de datos de los que no pueden crearse instancias son: ST_Geometry, ST_Curve, ST_Surface, ST_MultiSurface y ST_MultiCurve.

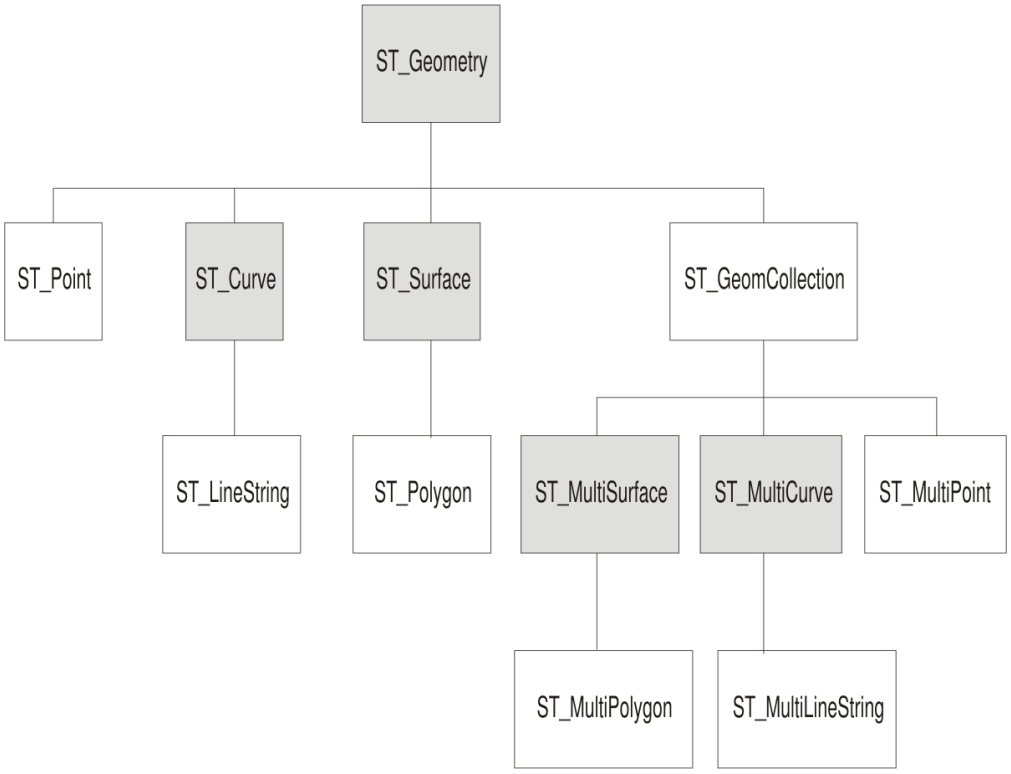


Figura 12. Jerarquía de tipos de datos espaciales. De los tipos de datos que aparecen en los cuadros blancos pueden crearse instancias. De los tipos de datos que aparecen en los cuadros sombreados no pueden crearse instancias.

La jerarquía de la Figura 12 incluye:

- Tipos de datos para elementos geográficos que se pueden percibir como formados por una sola unidad; por ejemplo, edificios individuales y lagos aislados.
- Tipos de datos para elementos geográficos que constan de varias unidades o componentes; por ejemplo, sistemas de canales o grupos de islas en un lago.
- Un tipo de datos para elementos geográficos de todo tipo.

Tipos de datos para elementos geográficos de una sola unidad

Utilice ST_Point, ST_LineString y ST_Polygon para almacenar coordenadas que definan el espacio ocupado por elementos que se pueden percibir como una sola unidad:

- Utilice ST_Point cuando desee indicar el punto en el espacio ocupado por un elemento geográfico aislado. El elemento puede ser muy pequeño (un pozo de agua), muy grande (una ciudad) o de tamaño intermedio (un complejo de edificios o un parque). En cada caso, el punto en el espacio se puede situar 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 dato ST_Point incluye una coordenada X y una coordenada Y que definen una intersección de este tipo. 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 definan el espacio ocupado por elementos lineales; por ejemplo, calles, canales y conductos.
- Utilice ST_Polygon cuando desee indicar la extensión de espacio ocupada por un elemento que conste de varios lados; por ejemplo, un barrio de una ciudad, un

bosque o un hábitat natural. Un dato de ST_Polygon consta de las coordenadas que definen el perímetro de un elemento de este tipo.

En algunos casos, se puede utilizar ST_Polygon y ST_Point para un mismo elemento. Por ejemplo, supongamos que necesita información espacial sobre un complejo de apartamentos. Si desea representar el punto en el espacio en el que se encuentra cada edificio en el complejo, utilizaría ST_Point para guardar las coordenadas X e Y que definen ese punto. De lo contrario, si desea representar el área total ocupada por el complejo, utilizaría ST_Polygon para guardar las coordenadas que definen el perímetro de esta zona.

Tipos de datos para elementos geográficos formados por varias unidades

Utilice ST_MultiPoint, ST_MultiLineString y ST_MultiPolygon para almacenar coordenadas que definen espacios ocupados por elementos geográficos formados por varias unidades:

- Utilice ST_MultiPoint para representar elementos geográficos formados de unidades cuyas ubicaciones estén referenciadas por una coordenada X y una coordenada Y. Por ejemplo, considere una tabla cuyas filas representan cadenas de islas. Se han identificado la coordenada X y la coordenada Y para cada isla. Si desea que la tabla incluya estas coordenadas y las coordenadas para cada cadena en su totalidad, defina una columna ST_MultiPoint para albergar estas coordenadas.
- Utilice ST_MultiLineString cuando esté representado elementos compuestos de unidades lineales y desee almacenar las coordenadas para las ubicaciones de estas unidades y la ubicación para cada elemento en su totalidad. Por ejemplo, imagine una tabla cuyas filas representen sistemas fluviales. Si desea que la tabla incluya coordenadas para las ubicaciones de los sistemas y los componentes de los mismos, defina una columna ST_MultiLineString para albergar estas coordenadas.
- Utilice ST_MultiPolygon cuando esté representando elementos compuestos de unidades que tengan varios lados y desee almacenar las coordenadas para las ubicaciones de estas unidades y la ubicación de cada elemento en su totalidad. Por ejemplo, imagine una tabla cuyas filas representen regiones rurales o las granjas en cada región. Si desea que la tabla incluya coordenadas para las ubicaciones de las regiones y las granjas, defina una columna ST_MultiPolygon para albergar estas coordenadas.

Las múltiples unidades que forman el elemento geográfico no son una serie de entidades individuales, sino que son un agregado de las partes que forma la entidad completa.

Un tipo de datos para todos los elementos geográficos

Puede utilizar ST_Geometry cuando no esté seguro de cuál de los otros tipos de datos debe utilizar. Puesto que ST_Geometry es la raíz de la jerarquía a la que pertenecen los demás tipos de datos, una columna ST_Geometry puede contener los mismos tipos de datos que se pueden almacenar en columnas de los demás tipos de datos.

Atención:

Si piensa utilizar el geocodificador proporcionado, DB2SE_USA_GEOCODER, para crear datos para una columna espacial, la columna debe ser de tipo ST_Point o ST_Geometry. Sin embargo, determinadas herramientas de visualización no dan soporte a columnas ST_Geometry, sino sólo a columnas a las que se ha asignado un subtipo apropiado de ST_Geometry.

Tareas relacionadas:

- "Registro de columnas espaciales" en la página 87
- "Creación de columnas espaciales" en la página 86

Creación de columnas espaciales

Esta tarea forma parte de la tarea más amplia de "Configuración de recursos espaciales para un proyecto." Después de elegir un sistema de coordenadas y determinar qué sistema de referencia espacial va a utilizar para sus datos, creará una columna espacial en una tabla existente o importará los datos en una nueva tabla.

Requisitos previos:

Antes de crear una columna espacial, su ID de usuario debe tener las autorizaciones necesarias sobre las sentencias DB2 SQL CREATE TABLE o ALTER TABLE. El ID de usuario debe tener como mínimo uno de los privilegios o autorizaciones siguientes:

- Autorización SYSADM o DBADM sobre la base de datos donde reside la tabla que contiene la columna
- Autorización CREATETAB sobre la base de datos y privilegio USE para el espacio de tabla, así como uno de los siguientes:
  - Autorización IMPLICIT_SCHEMA sobre la base de datos, si el esquema implícito o explícito del índice no existe
  - Privilegio CREATEIN sobre el esquema, si el nombre de esquema del índice hace referencia a un esquema existente
- Privilegio ALTER para la tabla que se debe modificar
- Privilegio CONTROL para la tabla que se debe modificar
- Privilegio ALTERIN para el esquema de la tabla

Procedimiento:

Puede proporcionar columnas espaciales a la base de datos de varias maneras:

- Utilice la sentencia CREATE TABLE de DB2 para crear una tabla y para incluir una columna espacial en la tabla.
- Utilice la sentencia ALTER TABLE de DB2 para añadir una columna espacial a una tabla existente.
- Utilice la ventana Crear columna espacial del Centro de control de DB2. Abra la ventana Columnas espaciales desde una tabla. Para obtener más información sobre cómo utilizar esta ventana, consulte la ayuda en línea.
- Si está importando datos espaciales desde un archivo de formas, utilice DB2 Spatial Extender para crear una tabla y para proporcionar a esta tabla una columna donde alojar los datos. Consulte el apartado "Importación de datos de formas a una tabla nueva o existente" en la página 90.
- Si está importando datos espaciales desde un archivo de transferencia SDE, utilice DB2 Spatial Extender para crear una tabla, para proporciona a esta tabla una columna donde alojar los datos y para hacer que la columna sea accesible a las herramientas de visualización. Consulte el apartado "Importación de datos de transferencia SDE a una tabla nueva o existente" en la página 92.

Siguiente tarea: "Registro de columnas espaciales" en la página 87

Tareas relacionadas:

- "Importación de datos de formas a una tabla nueva o existente" en la página 90
- "Importación de datos de transferencia SDE a una tabla nueva o existente" en la página 92
- "Registro de columnas espaciales" en la página 87
- "Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 141
- "Creación de una columna espacial"

Información relacionada:

- "Sentencia ALTER TABLE" en el manual *Consulta de SQL, Volumen 2*
- "Sentencia CREATE TABLE" en el manual *Consulta de SQL, Volumen 2*
- "Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos" en la página 131

Registro de columnas espaciales

Es posible que desee registrar una columna espacial en los siguientes casos:

- Acceso mediante herramientas de visualización

  Si desea que ciertas herramientas de visualización—por ejemplo, ArcExplorer for DB2— generen visualizaciones gráficas de los datos de una columna espacial, es necesario que se asegure de la integridad de los datos de la columna. Esto se hace imponiendo una restricción que requiere que todas las filas de la columna utilicen el mismo sistema de referencia espacial. Para imponer esta restricción, registre la columna, especificando su nombre y el sistema de referencia espacial que se aplica a la misma.
- Acceso mediante índices espaciales

  Utilice el mismo sistema de coordenadas para todos los datos de una columna espacial en la que desee crear un índice para asegurarse de que el índice espacial devuelva los resultados correctos. La columna espacial se registra para forzar que los datos utilicen el mismo sistema de referencia espacial y, como corresponde, el mismo sistema de coordenadas.

Requisitos previos:

Antes de registrar una columna espacial, el ID de usuario debe tener una de las siguientes formas de autorización:

- Autorización SYSADM o DBADM en la base de datos en que reside la tabla que contiene la columna.
- Privilegio CONTROL o ALTER sobre esta tabla.

Si está utilizando el procesador de línea de mandatos db2se o un programa de aplicación para importar datos desde un archivo de transferencia SDE, puede hacer que DB2 Spatial Extender cree y registre automáticamente una columna para contener los datos. En dicho caso, el ID de usuario debe tener autorización de SYSADM o DBADM sobre la base de datos.

Procedimiento:

Puede registrar una columna espacial de cualquiera de las maneras siguientes:

- Utilice las ventanas Columnas espaciales y Seleccionar sistema de referencia espacial del Centro de control de DB2 para registrar la columna.

- Emita el mandato **db2se register_spatial_column**.
- Ejecute una aplicación que invoque al procedimiento almacenado db2gse.ST_register_spatial_column.
- Si desea importar datos espaciales desde un archivo de transferencia SDE, puede utilizar la ventana "Importar Datos Espaciales" del Centro de control, el mandato **import_sde** o el procedimiento almacenado db2gse.ST_import_sde para crear una tabla con una columna espacial, para registrar esta columna y para importar los datos a la columna.

Consulte la columna SRS_NAME en la vista DB2GSE.GSE_GEOMETRY_COLUMNS para comprobar el sistema de referencia espacial que ha seleccionado para una columna determinada después de registrar la columna.

Tareas relacionadas:

- "Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 141

Información relacionada:

- "Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos" en la página 131
- "ST_register_spatial_column" en la página 279

Capítulo 10. Llenado de columnas espaciales

Después de crear columnas espaciales, y registrar las que se deberán acceder mediante herramientas de visualización, está preparado para llenar las columnas con datos espaciales. Existen tres maneras de proporcionar los datos: importarlos; utilizar un geocodificador para obtenerlos de los datos comerciales; o bien utilizar funciones espaciales para crearlos o para obtenerlos de los datos comerciales o de otros datos espaciales. Este capítulo:

- Trata acerca del concepto y de la tarea de importar datos espaciales a la base de datos y del concepto y de la tarea de exportar datos espaciales a archivos que las aplicaciones pueden utilizar.
- Trata acerca de la geocodificación y presenta las tareas de configurar las operaciones de geocodificación, configurar los geocodificadores para que se ejecuten automáticamente y ejecutar los geocodificadores en modalidad de proceso por lotes.

Cómo importar y exportar datos espaciales

Esta sección trata acerca del concepto de importar y exportar datos y presenta las tareas siguientes:

- Importación de datos espaciales a una nueva tabla o a una tabla o vista existente
- Exportación de datos espaciales a archivos que las aplicaciones pueden utilizar

Acerca de la importación y la exportación de datos espaciales

Puede utilizar DB2® Spatial Extender para intercambiar datos espaciales entre la base de datos y fuentes de datos externas. Más concretamente, puede importar datos espaciales desde fuentes externas transfiriéndolos a la base de datos en archivos, denominados *archivos de intercambio de datos*. También puede exportar datos espaciales desde la base de datos a archivos de intercambio de datos desde donde fuentes externas pueden adquirirlos. Esta sección sugiere algunas de las razones para importar y exportar datos espaciales y describe la naturaleza de los archivos de intercambio de datos que DB2 Spatial Extender soporta.

Razones para importar y exportar datos espaciales:

Importando datos espaciales, puede obtener mucha información espacial que ya está disponible en la empresa. Exportándolos, puede hacer que estén disponibles en un formato de archivo estándar para aplicaciones existentes. Supongamos los casos siguientes:

- La base de datos contiene datos espaciales que representan oficinas de ventas, clientes y otros elementos de 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 pueden obtenerse de un proveedor de mapas. Puede utilizar DB2 Spatial Extender para importarlos desde un archivo de intercambio de datos que suministra el proveedor.
- Desea migrar los datos espaciales desde un sistema Oracle al entorno DB2. Puede continuar utilizando el programa de utilidad de Oracle para grabar 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.

- No está conectado a DB2 y desea utilizar un geonavegador para mostrar presentaciones visuales de información espacial a los clientes. El navegador 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 navegador para producir los datos en formato visual.

Archivos de forma y archivos de transferencia SDE:

DB2 Spatial Extender da soporte a dos tipos de archivos de intercambio: archivos de formas y archivos de transferencia SDE. En realidad, el término *archivos de formas* designa un conjunto de archivos que tienen el mismo nombre de archivo pero diferentes extensiones de archivo. Este grupo puede incluir hasta cuatro archivos. Estas funciones son:

- Un archivo que contiene datos espaciales en *formato de forma*, un formato estándar de facto de la industria desarrollado por ESRI. Estos datos se denominan a menudo *datos de formas*. La extensión de un archivo que contiene datos de formas es .shp.
- Un archivo que contiene datos comerciales pertenecientes a ubicaciones definidas por datos de formas. La extensión de este archivo es .dbf.
- Un archivo que contiene un índice para datos de formas. La extensión de este archivo es .shx.
- Un archivo que contiene una especificación del sistema de coordenadas en el que se basan los datos en un archivo .shp. La extensión de este archivo es .prj.

Los archivos de formas se suelen utilizar para importar datos procedentes de sistemas de archivos y para exportar datos a archivos en sistemas de archivos.

Cuando utiliza DB2 Spatial Extender para importar datos de formas, recibe como mínimo un archivo .shp. En la mayoría de los casos, recibirá también uno o más de los otros tres tipos de archivos de formas.

Los *Archivos de transferencia SDE* se suelen utilizar para importar datos que se originan en bases de datos ESRI. Cada archivo incluye datos espaciales, un sistema de referencia espacial para estos datos y datos comerciales. Los datos espaciales, cuyo formato es propiedad de ESRI, están destinados a una columna de tabla que se ha registrado en el catálogo de DB2 Spatial Extender. Los datos comerciales están destinados a otras columnas de la tabla a la que pertenece la columna registrada.

Importación de datos espaciales

Esta sección proporciona una visión general de las tareas de importar archivos de formas y archivos de transferencia SDE a la base de datos. La sección incluye referencias cruzadas a información detallada que necesita saber (por ejemplo, procesos y parámetros) para poder realizar estas tareas.

Importación de datos de formas a una tabla nueva o existente

Puede importar datos de formas a una tabla o vista existente, o puede crear una tabla e importar los datos de formas a dicha tabla en una sola operación. Más específicamente, puede:

- Importar los datos de formas a una columna espacial en una tabla existente, una vista actualizable existente, o una vista existente en la que está definido un desencadenante INSTEAD OF para las operaciones INSERT

- Crear automáticamente una tabla con una columna espacial e importar los datos de formas a esta columna

Requisitos previos:

Antes de importar datos de formas a una tabla o vista existente, el ID de usuario debe tener una de las siguientes formas de autorización:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla o la vista
- Privilegio CONTROL sobre la tabla o la vista
- Privilegio INSERT sobre la tabla o la vista
- Privilegio SELECT sobre la tabla o la vista (necesario sólo si la tabla incluye una columna de ID que no es una columna IDENTITY)
- Privilegios para acceder a los directorios a los que pertenecen los archivos de entrada y los archivos de error
- Privilegios de lectura sobre los archivos de entrada y privilegios de escritura sobre los archivos de error

Antes de empezar a crear automáticamente una tabla e importar los datos de formas a la misma, el ID de usuario debe tener las siguientes formas de autorización:

- Autorización SYSADM, DBADM o CREATETAB sobre la base de datos que contiene la tabla
- Uno de los permisos siguientes:
  - Privilegio CREATEIN sobre el esquema al que pertenece la tabla (necesario cuando el esquema ya existe)
  - Autorización IMPLICIT_SCHEMA sobre la base de datos que contiene la tabla (necesario cuando el esquema especificado para la tabla no existe realmente)
- Privilegios para acceder a los directorios a los que pertenecen los archivos de entrada y los archivos de error
- Privilegios de lectura sobre los archivos de entrada y privilegios de escritura sobre los archivos de error

Procedimiento:

Puede importar datos de formas de cualquiera de las maneras siguientes:

- Utilice la ventana Importar datos de formas del Centro de control de DB2.
- Emita el mandato **db2se import_shape**.
- Ejecute una aplicación que llame al procedimiento almacenado db2gse.ST_import_shape.

Recomendación:

Puede mejorar el rendimiento del proceso de importación aprovechando funciones existentes en DB2. Por ejemplo, cuando importe datos a una tabla existente o recién creada, defina la tabla como NOT LOGGED INITIALLY especificando los parámetros apropiados de creación de tablas.

Conceptos relacionados:

- "Acerca de la importación y la exportación de datos espaciales" en la página 89

Tareas relacionadas:

- "Importación de datos de transferencia SDE a una tabla nueva o existente" en la página 92
- "Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 141

Información relacionada:

- "Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos" en la página 131
- "ST_import_shape" en la página 266

Importación de datos de transferencia SDE a una tabla nueva o existente

Puede importar datos de transferencia SDE a una tabla existente, o puede crear una tabla e importar los datos de transferencia SDE a dicha tabla en una sola operación. Más específicamente, puede:

- Importar datos de transferencia SDE a una tabla existente que incluya una columna espacial que ya esté registrada en el catálogo de DB2 Spatial Extender. Los datos de transferencia pueden incluir datos espaciales para la columna y datos comerciales para otras columnas de la tabla.
- Crear automáticamente una tabla que no tenga una columna espacial, registrar esta columna en el catálogo e importar datos de transferencia SDE a esta columna así como a las otras columnas de la tabla.

Requisitos previos:

Antes de importar datos a una columna de una tabla o vista existente, el ID de usuario debe tener una de las siguientes formas de autorización:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla o la vista
- Privilegio CONTROL sobre la tabla o la vista
- Los privilegios INSERT y SELECT sobre la tabla o la vista

Antes de iniciar la operación de crear una tabla automáticamente e importar los datos de formas a la misma, el ID de usuario debe tener las siguientes formas de autorización:

- Autorización SYSADM, DBADM o CREATETAB sobre la base de datos que contiene la tabla
- Uno de los permisos siguientes:
  - Privilegio CREATEIN sobre el esquema al que pertenece la tabla (necesario cuando el esquema ya existe)
  - Autorización IMPLICIT_SCHEMA sobre la base de datos que contiene la tabla (necesario cuando el esquema especificado para la tabla no existe realmente)

Procedimiento:

Puede importar datos de transferencia SDE de cualquiera de las maneras siguientes:

- Utilice la ventana Importar del Centro de control de DB2.
- Emita el mandato **db2se import_sde**.
- Ejecute una aplicación que llame al procedimiento almacenado db2gse.GSE_import_sde.

Para obtener información acerca de cómo realizar estas acciones, consulte las fuentes que se listan en el apartado "Tareas afines" al final de esta explicación.

Conceptos relacionados:

- "Acerca de la importación y la exportación de datos espaciales" en la página 89

Tareas relacionadas:

- "Importación de datos de formas a una tabla nueva o existente" en la página 90
- "Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 141

Información relacionada:

- "Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos" en la página 131
- "GSE_import_sde" en la página 236

Exportación de datos espaciales

Esta sección proporciona una visión general de las tareas de exportar datos espaciales a archivos de formas y de transferencia SDE. La sección incluye referencias cruzadas a información detallada que necesita saber (por ejemplo, procesos y parámetros) para poder realizar estas tareas.

Exportación de datos a un archivo de formas

Puede exportar los datos espaciales devueltos en los resultados de una consulta a un archivo de formas. Los datos pueden proceder de fuentes tales como una tabla, una unión de varias tablas, conjuntos de resultados devueltos al consultar vistas, o de datos de salida de una función espacial.

Si existe un archivo al que desea exportar datos, DB2 Spatial Extender puede añadir los datos a este archivo. Si dicho archivo no existe, puede utilizar DB2 Spatial Extender para crear uno.

Requisitos previos:

Para poder exportar datos a un archivo de formas, su ID de usuario debe tener los privilegios siguientes:

- Privilegio para ejecutar una subselección que devuelve los resultados que desea exportar
- Privilegio para escribir en el directorio donde reside el archivo al que exportará los datos
- Privilegio para crear un archivo donde colocar los datos exportados (necesario si dicho archivo no existe aún)

Para saber qué son estos privilegios y cómo obtenerlos, consulte con el administrador de la base de datos.

Procedimiento:

Puede exportar datos a un archivo de formas de cualquiera de las maneras siguientes:

- Inicialice la exportación desde la ventana Exportar archivo de formas del Centro de control de DB2.

- Emita el mandato **db2se export_shape** desde el procesador de línea de mandatos de db2se.
- Ejecute una aplicación que llame al procedimiento almacenado db2gse.ST_export_shape.

Para obtener información acerca de cómo realizar estas acciones, consulte las fuentes que se listan en el apartado "Tareas afines" al final de esta explicación.

Exportación de datos a un archivo de transferencia SDE

Puede exportar una tabla que contiene datos espaciales a un archivo de transferencia SDE. La tabla no puede contener más de una columna espacial. Además, esta columna debe estar registrada en el catálogo de DB2 Spatial Extender. Si la tabla contiene datos comerciales, estos datos se exportarán junto con los datos espaciales. Puede exportar todas las filas de la tabla o un subconjunto de filas. Para exportar un subconjunto, especifique una cláusula WHERE que identifique el subconjunto.

Requisitos previos:

Para poder exportar los datos a un archivo de transferencia SDE, su ID de usuario debe tener los permisos siguientes:

- Autorización SYSADM o DBADM.
- Privilegio SELECT sobre la tabla que se debe exportar.
- Privilegio para escribir en el directorio donde reside el archivo al que exportará los datos

Restricciones:

- Puede exportar sólo una columna espacial en cada operación de exportación.
- Las columnas que exporte deben tener tipos de datos soportados por el formato SDE.
- La tabla debe contener exactamente una columna espacial.
- Esta columna debe estar registrada en el catálogo de DB2 Spatial Extender.
- No puede realizar adiciones a archivos SDE existentes.

Procedimiento:

Puede exportar datos espaciales y comerciales a un archivo de transferencia SDE de cualquiera de las maneras siguientes:

- Utilice la ventana Exportar archivos del Centro de control de DB2.
- Emita el mandato **db2se export_sde**.
- Ejecute una aplicación que llame al procedimiento almacenado db2gse.GSE_export_sde.

Conceptos relacionados:

- "Acerca de la importación y la exportación de datos espaciales" en la página 89

Tareas relacionadas:

- "Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 141

Información relacionada:

- "Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos" en la página 131
- "GSE_export_sde" en la página 234

Cómo utilizar un geocodificador

Esta sección trata acerca del concepto de la geocodificación y presenta las tareas siguientes:

- Definición del trabajo que desea que realice un geocodificador; por ejemplo, especificación de cuántos registros debe procesar el geocodificador antes de enviar una confirmación
- Configuración de un geocodificador para geocodificar datos tan pronto como se añadan los datos a la tabla o se actualicen en la misma.
- Ejecución de un geocodificador en modalidad de proceso por lotes

Geocodificadores y geocodificación

Los términos *geocodificador* y *geocodificación* se utilizan en varios contextos. Esta explicación clasifica estos contextos, de forma que los significados de los términos queden claros cada vez que se encuentre con uno de ellos. La explicación define *geocodificador* y *geocodificación*, describe las modalidades en las que funciona un geocodificador, describe una actividad más amplia a la que pertenece la geocodificación, y resume las tareas de los usuarios que pertenecen a la geocodificación.

En DB2® Spatial Extender, un geocodificador es una función escalar que convierte datos existentes (los datos de entrada de la función) a datos que el usuario puede entender en términos espaciales (los datos de salida de la función). Normalmente, los datos existentes son datos relacionales que describen o nombran una ubicación. Por ejemplo, el geocodificador que se proporciona con DB2 Spatial Extender, DB2SE_USA_GEOCODER, convierte direcciones de los Estados Unidos a datos ST_Point. DB2 Spatial Extender también puede dar soporte a geocodificadores proporcionados por proveedores y a geocodificadores proporcionados por usuarios; y las necesidades de los datos de entrada y de salida de los mismos puede que no sean las mismas que con DB2SE_USA_GEOCODER. Para poner un ejemplo: Es posible que un geocodificador proporcionado por un proveedor convierta las direcciones en coordenadas que DB2 no almacene, sino que las grabe en un archivo. Es posible que otro pueda convertir el número de una oficina en un edificio comercial a coordenadas que definan la ubicación de la oficina en el edificio, o que pueda convertir un identificador de una estantería en un almacén a coordenadas que definan la ubicación de la estantería en el almacén.

En otros casos, es posible que los datos existentes que un geocodificador convierta sean datos espaciales. Por ejemplo, es posible que un geocodificador proporcionado por un usuario convierta coordenadas X e Y a datos que se adecuen a uno de los tipos de datos de DB2 Spatial Extender.

En DB2 Spatial Extender, *geocodificación* es simplemente la operación por la que el geocodificador convierte sus datos de entrada en datos de salida—convierte las direcciones en coordenadas, por ejemplo.

Modalidades:

Un geocodificador se puede ejecutar en dos modalidades:

- En *modalidad de proceso por lotes*, un geocodificador intenta, en una sola operación, convertir todos sus datos de entrada desde una sola tabla. Por ejemplo, en modalidad de proceso por lotes, DB2SE_USA_GEOCODER intenta convertir todas las direcciones en una sola tabla (o, alternativamente, todas las direcciones en un subconjunto especificado de filas de la tabla).
- En *modalidad automática*, un geocodificador convierte los datos tan pronto como se ha insertado o actualizado en una tabla. El geocodificador se activa mediante los desencadenantes de inserción y actualización que están definidos en la tabla.

Procesos de geocodificación:

La geocodificación es una de las varias operaciones por las que el contenido de una columna espacial de una tabla de DB2 se obtiene a partir de otros datos. Esta explicación hace referencia a estas operaciones en conjunto como un *proceso de geocodificación*. Los procesos de geocodificación pueden variar de un geocodificador a otro. Por ejemplo, DB2SE_USA_GEOCODER busca archivos de direcciones conocidas para determinar si cada dirección que recibe como dato de entrada coincide con una dirección conocida en un grado determinado. Debido a que las direcciones conocidas son como material de referencia que la gente buscará cuando realicen la búsqueda, estas direcciones se denominan colectivamente *datos de referencia.* Es posible que otros geocodificadores no necesiten datos de referencia; es posible que verifiquen sus datos de entrada de otras maneras. El proceso de geocodificación en el que DB2SE_USA_GEOCODER participa se realiza de la manera siguiente:

1. DB2SE_USA_GEOCODER realiza operaciones que se han diseñado para llevar a cabo:
   a. DB2SE_USA_GEOCODER analiza cada dirección que recibe como datos de entrada.
   b. DB2SE_USA_GEOCODER busca los datos de referencia para nombres de calle que, en un grado determinado, se parecen al nombre de calle en la dirección analizada. Restringe su búsqueda a calles en el área designada por el código postal de la dirección.
   c. Si la búsqueda es satisfactoria, DB2SE_USA_GEOCODER determina si alguna dirección de las calles que ha encontrado coincide en un grado determinado con la dirección analizada.
   d. Si DB2SE_USA_GEOCODER encuentra una coincidencia, geocodifica la dirección analizada. De lo contrario, devuelve un valor nulo.
2. Si DB2SE_USA_GEOCODER geocodifica la dirección analizada, DB2 coloca las coordenadas resultantes en una columna espacial designada.
3. Si DB2SE_USA_GEOCODER está geocodificando en modalidad de proceso por lotes, DB2 Spatial Extender emite una confirmación (a) cada vez que DB2SE_USA_GEOCODER acaba de procesar cierto número de registros de entrada o (b) después que DB2SE_USA_GEOCODER acabe de procesar todos sus datos de entrada.

Tareas del usuario:

En DB2 Spatial Extender, las tareas que pertenecen a la geocodificación son:

- Prescripción de cómo algunas partes del proceso de geocodificación se deben ejecutar para una columna espacial determinada; por ejemplo, estableciendo el grado mínimo en el que deben coincidir los nombres de calle en los registros de entrada y los nombres de calle en los datos de referencia, estableciendo el grado mínimo en el que las direcciones en los registros de entrada y las direcciones en los datos de referencia deben coincidir, y determinando cuántos registros se

deben procesar antes de cada confirmación. Se puede hacer referencia a esta tarea como *configuración de la geocodificación* o *configuración de las operaciones de geocodificación*.

- Especificación de que los datos se deben geocodificar automáticamente cada vez que se añaden a la tabla o se actualizan en la misma. Cuando se produce la geocodificación automática, tendrán lugar las instrucciones que el usuario ha especificado cuando configuró las operaciones de geocodificación (excepto para las instrucciones que impliquen confirmaciones; éstas sólo se aplican a la geocodificación de proceso por lotes). Esta tarea se denomina *configuración de un geocodificador para que se ejecute automáticamente.*
- Ejecución de un geocodificador en modalidad de proceso por lotes. Si el usuario ya ha establecido las operaciones de geocodificación, sus instrucciones seguirán siendo efectivas durante cada sesión de proceso por lotes, a menos que el usuario las altere temporalmente. Si el usuario no ha establecido las operaciones de geocodificación antes de una sesión determinada, el usuario puede especificar que éstas sean efectivas para esta sesión en concreto. Se puede hacer referencia a esta tarea como *ejecución de un geocodificador en modalidad de proceso por lotes* y *ejecución de la geocodificación en modalidad de proceso por lotes.*

Configuración de las operaciones de geocodificación

DB2 Spatial Extender le permite definir, de antemano, el trabajo que es necesario realizar cuando se invoca a un geocodificador. Por ejemplo, puede especificar:

- Para qué columna el geocodificador va a proporcionar datos.
- Si los datos de entrada que el geocodificador lee de la tabla o vista deben estar limitados a un subconjunto de filas de la tabla o vista.
- El rango o número de registros que el geocodificador debe geocodificar en sesiones de proceso por lotes dentro de una unidad de trabajo.
- Requisitos para operaciones específicas del geocodificador. Por ejemplo, DB2SE_USA_GEOCODER puede geocodificar sólo los registros que coinciden con los elementos correspondientes a los mismos en los datos de referencia en un grado especificado o superior. Este grado se denomina *grado mínimo de coincidencia.*

Debe especificar los parámetros descritos anteriormente antes de configurar el geocodificador para que se ejecute en modalidad automática. De entonces en adelante, cada vez que invoque al geocodificador (no sólo automáticamente, sino también en ejecuciones de proceso por lotes), se realizarán operaciones de geocodificación según las especificaciones. Por ejemplo, si especifica que se deben geocodificar 45 registros en modalidad de proceso por lotes dentro de cada unidad de trabajo, se emitirá una confirmación después de que se geocodifique cada cuadragésimo quinto registro. (Excepción: puede alterar temporalmente las especificaciones para sesiones individuales de geocodificación de proceso por lotes.)

No tiene que establecer valores por omisión para las operaciones de geocodificación antes de ejecutar el geocodificador en modalidad de proceso por lotes. En su lugar, en el momento de iniciar una sesión de proceso por lotes, puede especificar cómo se deben realizar las operaciones durante la duración de la ejecución. Si establece valores por omisión para las sesiones de proceso por lotes, puede alterarlos temporalmente, si es necesario, para sesiones individuales.

Requisitos previos:

Antes de establecer las operaciones de geocodificación para un geocodificador determinado, el ID de usuario debe tener una de las siguientes formas de autorización:

- Autorización SYSADM o DBADM sobre la base de datos que contiene las tablas sobre las que operará el geocodificador
- Privilegio SELECT y privilegio CONTROL o UPDATE sobre cada tabla sobre la que opera el geocodificador

Procedimiento:

Puede configurar operaciones de geocodificación de cualquiera de las maneras siguientes:

- Invóquelo desde la ventana Configurar geocodificación del Centro de control de DB2.
- Emita el mandato **db2se setup_gc**.
- Ejecute una aplicación que llame al procedimiento almacenado db2gse.ST_setup_geocoding.

Para obtener información acerca de cómo realizar estas acciones, consulte las fuentes que se listan en el apartado “Tareas afines” al final de esta explicación.

Recomendaciones:

- Cuando DB2SE_USA_GEOCODER lee un registro de un dato de dirección, intenta hacer coincidir dicho registro con un elemento correspondiente en los datos de referencia. De forma esquematizada, la forma en que este proceso tiene lugar es la siguiente: Primero, busca el dato de referencia de las calles cuyo código postal es el mismo que el código postal del registro. Si encuentra un nombre de calle que es similar al nombre en el registro en un cierto grado mínimo, o en un grado superior a este grado mínimo, continúa buscando una dirección completa. Si encuentra una dirección completa que es similar a la dirección en el registro en un cierto grado mínimo, o en un grado superior a este grado mínimo, geocodifica el registro. Si no encuentra una dirección de este tipo, devuelve un valor nulo.

  El grado mínimo en el que los nombres de calle deben coincidir se denomina *sensibilidad ortográfica*. El grado mínimo para el que todas las direcciones deben coincidir se denomina *grado mínimo de coincidencia*. Por ejemplo, si la sensibilidad ortográficas es de 80, la coincidencia entre los nombres de calle debe ser exacta como mínimo en un 80 por ciento para que el geocodificador busque la dirección completa. Si el grado mínimo de coincidencia es 60, la coincidencia entre las direcciones debe ser exacta como mínimo en un 60 por ciento para que el geocodificador geocodifique el registro.

  Puede especificar cuál debe ser la sensibilidad ortográfica y el grado mínimo de coincidencia en las búsquedas. Tenga en cuenta que es posible que necesite ajustar estos valores. Por ejemplo, supongamos que la sensibilidad ortográfica y el grado mínimo de coincidencia sean ambas de 95. Si las direcciones que desea geocodificar no se han validado cuidadosamente, es altamente improbable que haya coincidencias con el 95 por ciento de exactitud. Como resultado, es probable que el geocodificador devuelva un valor nulo cuando procese estos registros. En dicho caso, es aconsejable que disminuya la sensibilidad y el grado mínimo de coincidencia, y que ejecute de nuevo el geocodificador. Los valores recomendados para el grado de sensibilidad ortográfica y el grado mínimo de coincidencia son 70 y 60, respectivamente.

- Como se ha indicado al principio de esta explicación, puede determinar si los datos de entrada que lee el geocodificador de la tabla o vista se deben limitar a un subconjunto de filas de la tabla o vista. Por ejemplo, suponga los siguientes casos:
  - Invoca al geocodificador para geocodificar direcciones de una tabla en modalidad de proceso por lotes. Desafortunadamente, el grado mínimo de coincidencia es demasiado alto, lo que provoca que el geocodificador devuelva un valor nulo cuando procesa la mayoría de las direcciones. Reduzca el grado mínimo de coincidencia cuando ejecute de nuevo el geocodificador. Para limitar sus datos de entrada a las direcciones que no se han geocodificado, especifique que debe seleccionar sólo aquellas filas que contienen el valor nulo que ha devuelto anteriormente.
  - El geocodificador sólo selecciona filas que se han añadido después de una fecha determinada.
  - El geocodificador selecciona sólo las filas que contienen direcciones en un área determinada; por ejemplo, un conjunto de regiones o una provincia.
- Como se ha indicado al inicio de esta explicación, puede determinar el número de registros que el geocodificador debe procesar en sesiones de proceso por lotes dentro de una unidad de trabajo. Puede hacer que el geocodificador procese el mismo número de registros en cada unidad de trabajo, o bien puede hacer que procese todos los registros de una tabla dentro de una única unidad de trabajo. Si elige esta última alternativa, tenga en cuenta lo siguiente:
  - Tiene menos control sobre el tamaño de la unidad de trabajo de lo que le permite la alternativa anterior. En consecuencia, no puede controlar cuántos bloqueos se mantienen o cuántas entradas de anotaciones cronológicas se realizan cuando opera el geocodificador.
  - Si el geocodificador encuentra un error que necesita una retrotracción, es necesario ejecutar de nuevo el geocodificador para que se ejecute para todos los registros. El consiguiente consumo de recursos puede ser alto si la tabla es muy grande y el error y la cancelación de cambios se producen una vez procesados la mayoría de los registros.

Configuración de un geocodificador para que se ejecute automáticamente

Puede configurar un geocodificador para que convierta automáticamente los datos a medida que éstos datos se añadan a una tabla o se actualicen en ella.

Requisitos previos:

Antes de poder configurar un geocodificador para que se ejecute automáticamente:

- Debe configurar las operaciones de geocodificación para cada columna espacial que se va a llenar con los datos de salida de un geocodificador.
- El ID de usuario debe tener las siguientes formas de autorización:
  - Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla en la que se definirán los desencadenantes para invocar al geocodificador
  - Uno o varios privilegios de esta tabla:
    - Privilegio CONTROL.
    - Si no tiene el privilegio CONTROL, necesita los privilegios ALTER, SELECT y UPDATE.
  - Los privilegios necesarios para crear desencadenantes en esta tabla.

Procedimiento:

Existen tres maneras de configurar la geocodificación automática:

- Hágalo desde la ventana Configurar geocodificación o desde la ventana Geocodificación del Centro de control de DB2.
- Emita el mandato **db2se enable_autogc**.
- Ejecute una aplicación que llame al procedimiento almacenado db2gse.ST_enable_autogeocoding.

Para obtener información acerca de cómo realizar estas acciones, consulte las fuentes que se listan en el apartado "Tareas afines" al final de esta explicación.

Recomendaciones:

- Puede configurar un geocodificador para que se ejecute automáticamente antes de invocarlo en modalidad de proceso por lotes. Por lo tanto, es posible que la geocodificación automática preceda a la geocodificación de proceso por lotes. Si esto sucede, es probable que la geocodificación de proceso por lotes invoque al proceso de los mismos datos que se han procesado automáticamente. Esta redundancia no causará datos duplicados, porque cuando los datos espaciales se producen dos veces, el segundo grupo de datos prevalece sobre el primero. Sin embargo, esto puede degradar el rendimiento.
- Antes de decidir si desea geocodificar los datos de direcciones de una tabla en modalidad de proceso por lotes o en modalidad automática, considere lo siguiente:
  - El rendimiento es mejor en la geocodificación de proceso por lotes que en la geocodificación automática. Una sesión de proceso por lotes se abre con una inicialización y finaliza con una limpieza. En la geocodificación automática, cada dato se geocodifica en una única operación que empieza con la inicialización y finaliza con la limpieza.
  - En conjunto, es probable que una columna espacial que se ha llenado mediante una geocodificación automática esté más actualizada que una columna espacial que se haya llenado mediante una geocodificación de proceso por lotes. Después de cada sesión de proceso por lotes, los datos de direcciones se pueden acumular y permanecer sin geocodificar hasta la siguiente sesión. Pero si ya geocodificación automática ya está habilitada, los datos de direcciones se geocodifican tan pronto como se almacenan en la base de datos.

Ejecución de un geocodificador en modalidad de proceso por lotes

Puede invocar a un geocodificador para que se ejecute en modalidad de proceso por lotes; es decir, que intente, en una sola operación, convertir varios registros en datos espaciales que van a ir a una columna específica.

En cualquier momento antes de ejecutar un geocodificador para que llene una columna espacial determinada, puede configurar operaciones de geocodificación para dicha columna. La configuración de las operaciones implica especificar cómo se van a cumplir ciertos requisitos cuando se ejecute el geocodificador. Por ejemplo, suponga que necesita que DB2 Spatial Extender emita una confirmación cada vez que el geocodificador procese 100 registros de entrada. Cuando haya configurado las operaciones, especificará 100 como el número necesario.

Cuando ya esté preparado para ejecutar el geocodificador, puede alterar temporalmente cualquiera de los valores que ha especificado cuando configuró las operaciones. Los valores con los que ha alterado temporalmente estos valores sólo serán efectivos durante la duración de la ejecución.

Si no ha configurado las operaciones, debe, cada vez que esté preparado para ejecutar el geocodificador, especificar cómo se cumplirán los requisitos durante la ejecución.

Requisitos previos:

Antes de ejecutar un geocodificador en modalidad de proceso por lotes, el ID de usuario debe tener una de las siguientes formas de autorización:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla cuyos datos se van a geocodificar
- Privilegio CONTROL o UPDATE sobre esta tabla
-

También necesita el privilegio SELECT sobre esta tabla, de forma que pueda especificar el número de registros que se procesarán antes de cada confirmación. Si especifica cláusulas WHERE para limitar las filas en las que operará el geocodificador, es posible que también necesite el privilegio SELECT sobre cualquier tabla y vista a la que haga referencia en dichas cláusulas. Consulte al administrador de la base de datos.

Restricciones:

Procedimiento:

Puede invocar a un geocodificador para que se ejecute en modalidad de proceso por lotes de cualquiera de las maneras siguientes:

- Invóquelo desde la ventana Ejecutar geocodificación del Centro de control de DB2.
- Emita el mandato **db2se run_gc**.
- Ejecute una aplicación que llame al procedimiento almacenado db2gse.ST_run_geocoding.

Llenado de columnas espaciales

Capítulo 11. Utilización de índices y vistas para acceder a datos espaciales

Antes de consultar columnas espaciales, puede crear índices y vistas que facilitarán el acceso a ellas. Este capítulo:

- Describe la naturaleza de los índices que Spatial Extender utiliza para activar el acceso a los datos espaciales
- Explica cómo crear estos índices
- Explica cómo utilizar las vistas para acceder a los datos espaciales

Tipos de índices espaciales

Para obtener un buen rendimiento, es importante tener índices eficientes definidos para la columnas de las tablas base de una base de datos. El rendimiento de la consulta está relacionado directamente con la velocidad con que se pueden encontrar los valores durante la consulta. Las consultas que utilizan un índice se puede ejecutar más rápidamente y proporcionan mejoras de rendimiento significativas.

Habitualmente, en las consultas espaciales intervienen dos o más dimensiones. Por ejemplo, una consulta espacial puede desear conocer si un punto está incluido dentro de un área (polígono). Debido a la naturaleza multidimensional de las consultas espaciales, la indexación por árbol binario nativo de DB2® es ineficaz para estas consultas.

Las consultas espaciales pueden utilizar los siguientes tipos de índice:

- Índices reticulares espaciales

  La tecnología de indexación de DB2 Spatial Extender utiliza la *indexación reticular*, que está diseñada para indexar datos espaciales multidimensionales y columnas espaciales. DB2 Spatial Extender proporciona un índice reticular que está optimizado para datos de dos dimensiones en una proyección plana de la tierra.
- Índices Voronoi geodésicos

  DB2 Geodetic Extender proporciona soporte para un nuevo método de acceso espacial que permite crear índices en columnas que contengan datos geodésicos multidimensionales. Un índice Voronoi geodésico es más apropiado que un índice reticular para los datos geodésicos porque considera la tierra como a una esfera continua sin distorsiones alrededor de los polos o bordes del meridiano 180º.

Conceptos relacionados:

- "Índices Voronoi geodésicos" en la página 183
- "Índices reticulares espaciales" en la página 104

Tareas relacionadas:

- "Creación de índices Voronoi geodésicos" en la página 187
- "Creación de índices reticulares espaciales" en la página 111

Información relacionada:

• "Funciones que utilizan índices para optimizar consultas" en la página 128

Índices reticulares espaciales

Los índices mejoran el rendimiento de las consultas de la aplicación, especialmente cuando la tabla o las tablas consultas contienen muchas filas. Si crea índices apropiados que el optimizador de consultas pueda elegir para ejecutar la consulta, reducirá de forma significativa el número de filas que se procesarán.

DB2 Spatial Extender proporciona un índice reticular optimizado para dos datos dimensionales. Este índice se crea en las dimensiones X e Y de una geometría.

Los siguientes aspectos de un índice reticular le ayudarán a comprender:

- La generación del índice
- El uso de funciones espaciales en una consulta
- Cómo utiliza una consulta un índice reticular espacial

Generación de índices reticulares espaciales

Spatial Extender genera un índice reticular espacial utilizando el rectángulo delimitador mínimo (MBR) de una geometría. Para muchas geometrías, el MBR es un rectángulo que rodea la geometría. Para obtener más detalles sobre los MBR, consulte el apartado "ST_MBR" en la página 442.

Un índice reticular espacial divide una región en retículas cuadradas lógicas con un tamaño fijo especificado por el usuario al crear el índice. El índice espacial se construye en una columna espacial creando una o más entradas para las intersecciones del MBR de cada geometría con las celdas reticulares. Un índice consta del identificador de celda reticular, del MBR de la geometría y del identificador interno de la fila que contiene la geometría.

Puede definir hasta tres niveles de índices espaciales (niveles reticulares). La utilización de varios niveles reticulares es ventajosa porque le permite optimizar el índice para diferentes tamaños de datos espaciales. Para obtener más información, consulte el apartado "Consideraciones sobre el número de niveles de índice y tamaños de retícula" en la página 106.

Si una geometría forma intersección con cuatro o más celdas reticulares, la geometría se promociona al siguiente nivel más grande. En general, las geometrías más grandes se indexarán en los niveles más grandes. Si una geometría forma intersección con 10 o más celdas reticulares en el tamaño de retícula más grande, se utiliza un nivel de índice de desbordamiento definido por el sistema. Este nivel de desbordamiento impide la generación de demasiadas entradas de índice. Para obtener el mejor rendimiento, defina tamaños de retícula para evitar el uso de este nivel de desbordamiento.

Por ejemplo, si hay varios niveles reticulares, el algoritmo de creación de índices intenta utilizar el nivel reticular más bajo posible para proporcionar la resolución más fina para los datos indexados. Cuando una geometría forma intersección con más de cuatro celdas reticulares en un nivel determinado, se promociona al nivel superior siguiente, (siempre que exista otro nivel). Por lo tanto, un índice espacial que tenga tres niveles reticulares de 10.0, 100.0 y 1000.0 formará primero intersección con cada geometría con retícula de nivel 10.0. Si una geometría forma intersección con más de cuatro celdas reticulares con un tamaño de 10.0, se promociona y forma intersección con la retícula de nivel 100.0. Si más de cuatro

intersecciones se producen en el nivel 100.0, la geometría se promociona al nivel 1000.0. Si más de 10 intersecciones se producen en el nivel 1000.0, la geometría se indexa en el nivel de desbordamiento.

Uso de funciones espaciales en una consulta

El optimizador de DB2 UDB puede utilizar un índice reticular espacial cuando una consulta contiene una de las siguientes funciones en su cláusula WHERE:

- ST_Contains
- ST_Crosses
- ST_Distance
- ST_EnvIntersects
- EnvelopesIntersect
- ST_Equals
- ST_Intersects
- ST_MBRIntersects
- ST_Overlaps
- ST_Touches
- ST_Within

Para obtener más información, consulte el apartado “Funciones que utilizan índices para optimizar consultas” en la página 128.

Cómo utiliza una consulta un índice reticular espacial

Cuando el optimizador de consultas elige un índice reticular espacial, la ejecución de la consulta utiliza el siguiente proceso de filtrado que consta de varios pasos:

1. Determinar las celdas reticulares que forman intersección con la ventana de consulta. La *ventana de consulta* es la geometría que le interesa y que especifica como segundo parámetro en una función espacial (consulte los ejemplos más adelante).
2. Explorar el índice para localizar entradas que tengan identificadores de celdas reticulares coincidentes.
3. Comparar los valores de MBR de la geometría en las entradas de índice con la ventana de consulta y descartar los valores que estén fuera de la ventana de consulta.
4. Realizar análisis adicionales si es preciso. Podrían realizarse análisis adicionales en el conjunto de geometrías candidato de los pasos anteriores para determinar si reúnen los requisitos de la función espacial (ST_Contains, ST_Distance, etcétera). La función espacial EnvelopesIntersect omite este paso y normalmente ofrece el mejor rendimiento.

Los siguientes ejemplos de consultas espaciales tienen un índice reticular espacial en la columna C.GEOMETRY:

```
SELECT nombre
FROM condados AS c
WHERE EnvelopesIntersect(c.geometry, -73.0, 42.0, -72.0, 43.0, 1) = 1

SELECT nombre
FROM condados AS c
WHERE ST_Intersects(c.geometry, :geometry2) = 1
```

En el primer ejemplo, los cuatro valores de coordenadas definen la ventana de consulta. Estos valores de coordenadas establecen las esquinas inferior izquierda y superior derecha 42.0–73.0 y 43.0 –72.0) de un rectángulo.

En el segundo ejemplo, Spatial Extender calcula el MBR de la geometría especificada por la variable del sistema principal :geometry2 y la utiliza como ventana de consulta.

Al crear un índice reticular espacial, deberá especificar los tamaños de retícula apropiados para los tamaños de ventana de consulta más habituales que suela utilizar su aplicación espacial. Si un tamaño de retícula es superior, deben explorarse las entradas de índice de las geometrías que estén fuera de la ventana de consulta, porque se encuentran en celdas reticulares que forman intersección con la ventana de consulta y dichas exploraciones adicionales disminuyen el rendimiento. Sin embargo, un tamaño reticular inferior podría generar más entradas de índice para cada geometría, por lo que deberán explorarse más entradas de índice y el rendimiento de la consulta también se verá afectado.

DB2 Spatial Extender proporciona el programa de utilidad Index Advisor, que analiza los datos de las columnas espaciales y recomienda tamaños de retícula apropiados para los tamaños de ventana de consulta habituales. Para obtener más información, consulte el apartado "Determinación de los tamaños de retícula para un índice reticular espacial" en la página 116.

Conceptos relacionados:

- "Consideraciones sobre el número de niveles de índice y tamaños de retícula" en la página 106
- "Tipos de índices espaciales" en la página 103
- "Ajuste de índices reticulares espaciales con Index Advisor—Visión general" en la página 115

Tareas relacionadas:

- "Creación de índices reticulares espaciales" en la página 111

Información relacionada:

- "Sentencia CREATE INDEX para un índice reticular espacial" en la página 113

Consideraciones sobre el número de niveles de índice y tamaños de retícula

Utilice Index Advisor para determinar tamaños de retícula apropiados para los índices reticulares espaciales, porque es la mejor manera de ajustar los índices y mejorar la eficacia de las consultas espaciales. Esta sección aclara conceptos que le ayudarán a entender los efectos de los diferentes niveles y tamaños de retícula.

Número de niveles reticulares

Puede utilizar hasta tres niveles reticulares. Sin embargo, para cada nivel reticular de un índice reticular espacial se realizará una búsqueda por índice separada durante una consulta espacial. Por lo tanto, si tiene más niveles reticulares, la consulta será menos eficaz.

Si los valores contenidos en la columna espacial tienen aproximadamente el mismo tamaño relativo, utilice un solo nivel reticular. Sin embargo, una columna espacial típica no contiene geometrías con el mismo tamaño relativo, aunque las geometrías de una columna espacial pueden agruparse de acuerdo con el tamaño. El usuario debe asociar los tamaños de retícula con estas agrupaciones de geometrías.

Por ejemplo, suponga que tiene una tabla con parcelas de terrenos en una columna espacial que contiene agrupaciones de pequeñas parcelas urbanas rodeadas por parcelas rurales mayores. Debido a que los tamaños de las parcelas se pueden agrupar en dos grupos (pequeñas parcelas urbanas y parcelas rurales mayores), especificaría dos niveles reticulares para el índice reticular espacial.

Tamaños de las celdas reticulares

La norma general es disminuir lo máximo posible los tamaños de retícula para obtener la resolución más alta y al mismo tiempo minimizar el número de entradas de índice.

- Se debe utilizar un valor pequeño para el tamaño de retícula más fino con el propósito de optimizar el índice general para las geometrías pequeñas de la columna. Esto evita el coste de evaluar geometrías que no están en el área de búsqueda. Sin embargo, el tamaño de retícula más fino también produce el mayor número de entradas de índice. En consecuencia, el número de entradas de índice procesadas durante el tiempo de la consulta aumenta, así como la cantidad de almacenamiento necesario para el índice. Estos factores reducen el rendimiento general.
- Mediante la utilización de tamaños de retícula más grandes, se puede optimizar más el índice para geometrías más grandes. Los tamaños de retícula más grandes producen menos entradas de índice para geometrías grandes que lo haría el tamaño de retícula más fino. En consecuencia, se reducen los requisitos de almacenamiento para el índice, aumentando el rendimiento general.

Las siguientes figuras muestran los efectos de los diferentes tamaños de retícula.

La Figura 13 en la página 108 muestra un mapa de parcelas de terreno, en el que cada parcela se representa por medio de una geometría de polígonos. El rectángulo negro representa una ventana de consulta. Supongamos que desea encontrar todas las geometrías cuyo MBR forma intersección con la ventana de consulta. La Figura 13 en la página 108 muestra que 28 geometrías (resaltadas en rosa) tienen un MBR que forma intersección con la ventana de consulta.

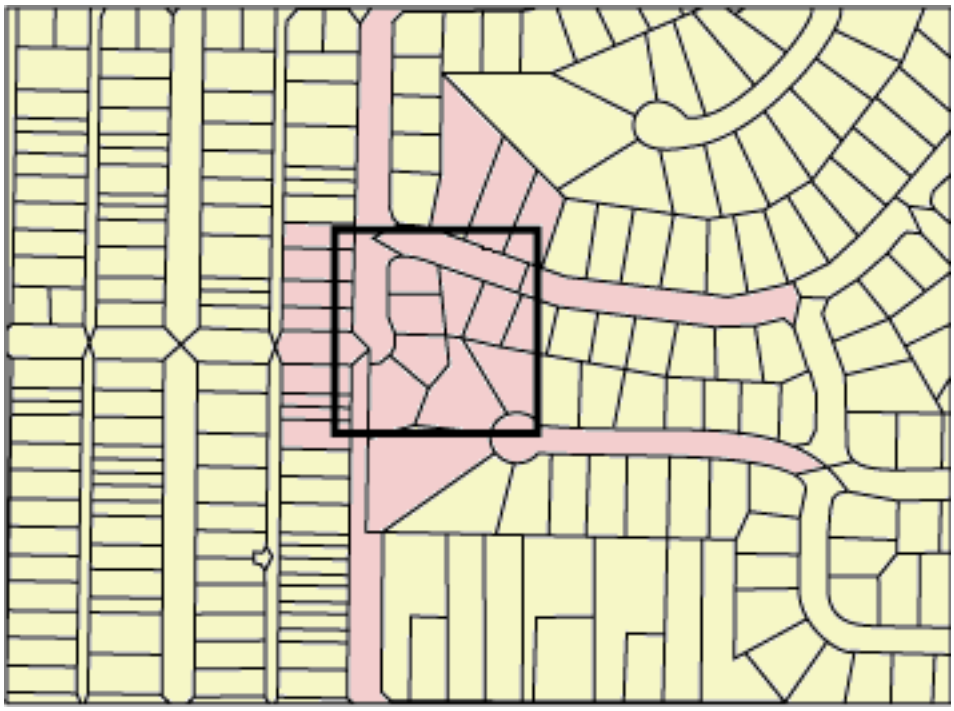


Figura 13. Parcelas de terreno en un vecindario

La Figura 14 en la página 109 muestra un tamaño de retícula pequeño (25) que se adapta mejor a la ventana de consulta.

- La consulta devuelve sólo las 28 geometrías que están resaltadas, pero debe examinar y descartar tres geometrías adicionales cuyos MBR forman intersección con la ventana de consulta.
- Este tamaño de retícula pequeño produce muchas entradas por geometría. Durante su ejecución, la consulta accede a todas las entradas de índice de estas 31 geometrías. La Figura 14 en la página 109 muestra 256 celdas reticulares que cubren la ventana de consulta. Sin embargo, la ejecución de la consulta accede a 578 entradas de índice porque muchas geometrías están indexadas con las mismas celdas reticulares.

Para esta ventana de consulta, este tamaño de retícula pequeño produce un número excesivo de entradas de índice para explorar.

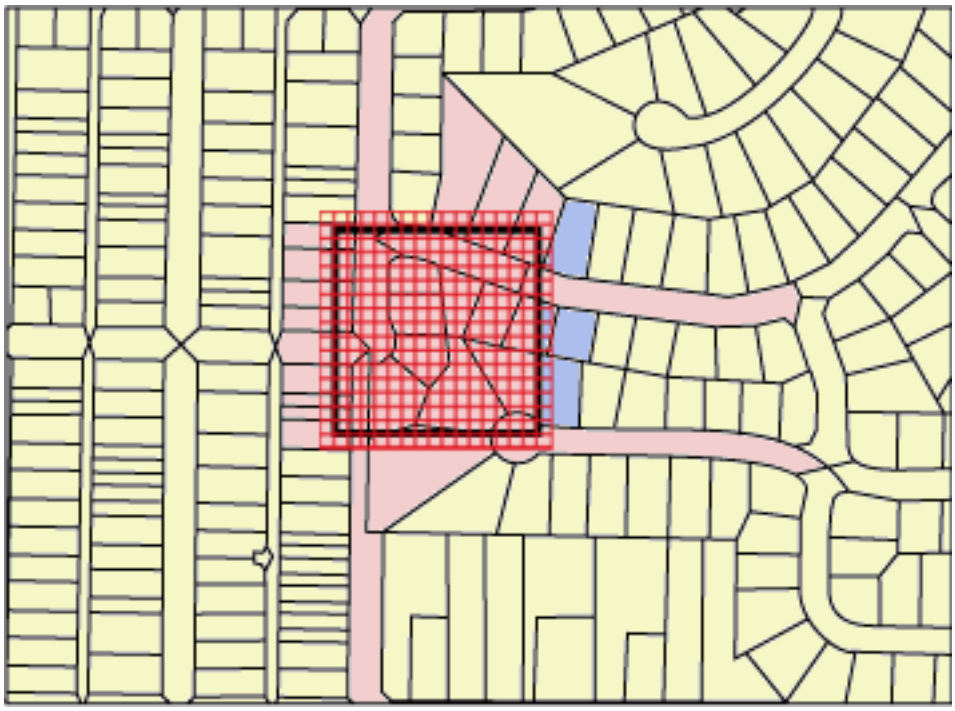


Figura 14. Tamaño de retícula pequeño (25) en parcelas de terreno

La Figura 15 en la página 110 muestra un tamaño de retícula grande (400) que abarca un área considerablemente mayor con muchas más geometrías que la ventana de consulta.

- Este tamaño de retícula grande produce sólo una entrada de índice por geometría, pero la consulta debe examinar y descartar 59 geometrías adicionales cuyos MBR forman intersección con la celda reticular.
- Durante su ejecución, la consulta accede a todas las entradas de índice para las 28 geometrías que forman intersección con la ventana de consulta, además de las entradas de índice para las 59 geometrías adicionales, para un total de 112 entradas de índice.

Para esta ventana de consulta, este tamaño de retícula grande produce un número excesivo de geometrías para examinar.

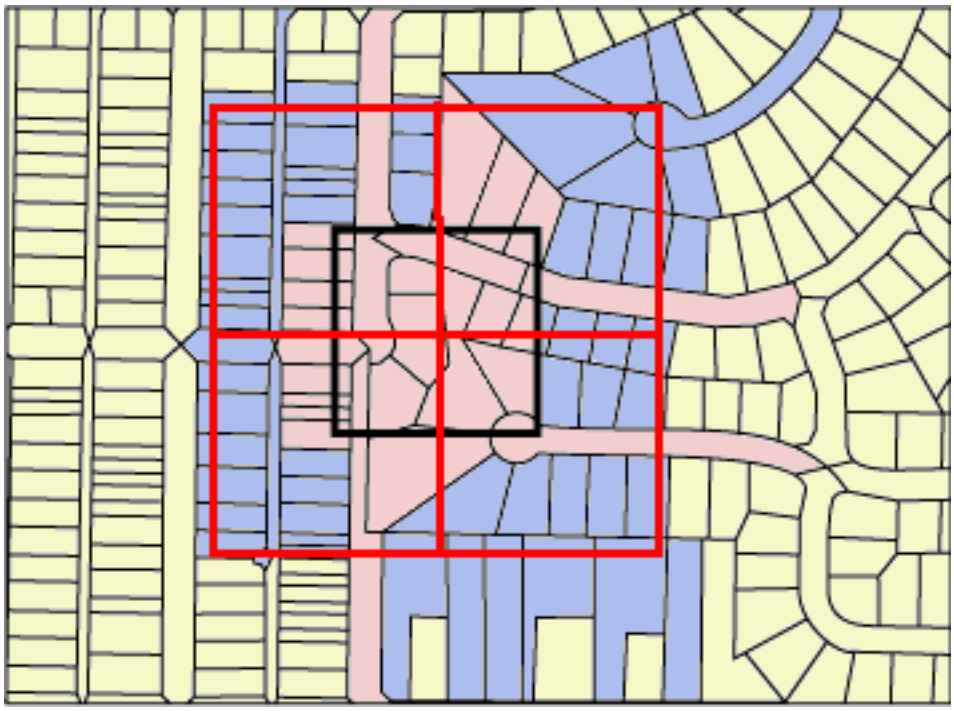


Figura 15. Tamaño de retícula grande (400) en parcelas de terreno

La Figura 16 en la página 111 muestra un tamaño de retícula medio (100) que se adapta mejor a la ventana de consulta.

- La consulta devuelve sólo las 28 geometrías que están resaltadas, pero debe examinar y descartar cinco geometrías adicionales cuyos MBR forman intersección con la ventana de consulta.
- Durante su ejecución, la consulta accede a todas las entradas de índice para las 28 geometrías que forman intersección con la ventana de consulta, además de las entradas de índice para las 5 geometrías adicionales, para un total de 91 entradas de índice.

Para esta ventana de consulta, este tamaño de retícula medio es el mejor porque produce un número significativamente inferior de entradas de índice que el tamaño de retícula pequeño y la consulta examina menos geometrías adicionales que el tamaño de retícula grande.

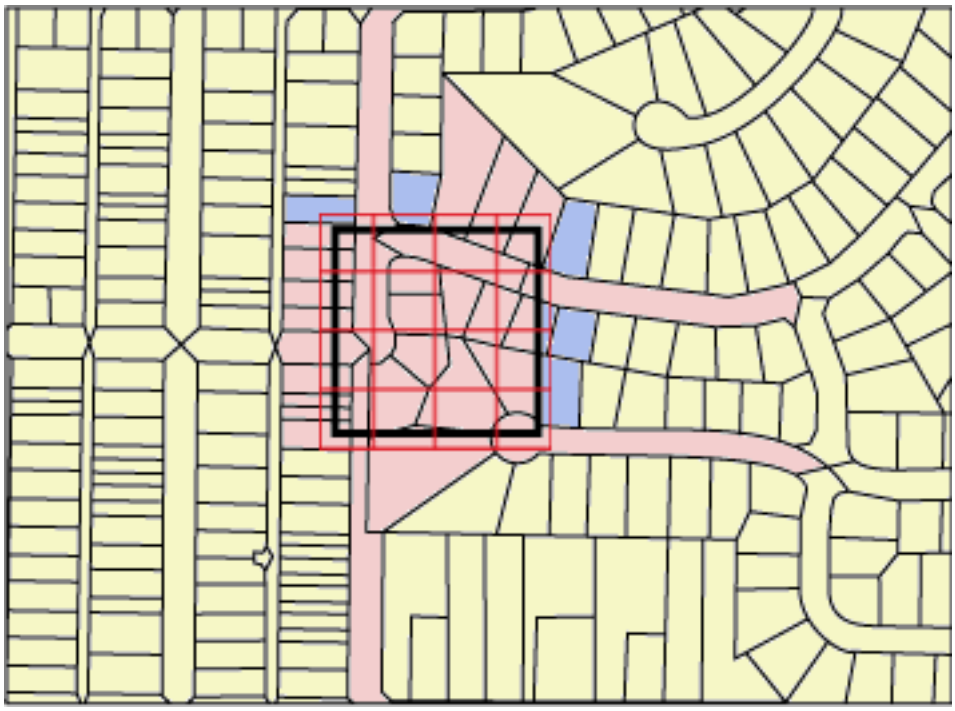


Figura 16. Tamaño de retícula medio (100) en parcelas de terreno

Conceptos relacionados:

- "Índices reticulares espaciales" en la página 104
- "Ajuste de índices reticulares espaciales con Index Advisor—Visión general" en la página 115

Tareas relacionadas:

- "Determinación de los tamaños de retícula para un índice reticular espacial" en la página 116
- "Análisis de estadísticas de índices reticulares espaciales" en la página 117
- "Creación de índices reticulares espaciales" en la página 111

Información relacionada:

- "Funciones que utilizan índices para optimizar consultas" en la página 128

Creación de índices reticulares espaciales

Los índices reticulares espaciales se crean para mejorar el rendimiento de las consultas en las columnas espaciales.

Cuando se crea un índice reticular espaciales se debe proporcionar la siguiente información:

- Un nombre
- El nombre de la columna espacial en la que se va a definir
- Los tamaños de retícula (consulte el apartado "Índices reticulares espaciales" en la página 104). La combinación de los tres tamaños de retícula ayuda a optimizar el rendimiento reduciendo el número total de entradas de índice y el número de las mismas que deben explorarse para satisfacer una consulta.

Requisitos previos:

Para crear un índice reticular espacial:

- Su ID de usuario debe tener las autorizaciones necesarias para la sentencia CREATE INDEX de SQL de DB2. El ID de usuario debe tener como mínimo uno de los siguientes privilegios o autorizaciones:
  - Autorización SYSADM o DBADM sobre la base de datos donde reside la tabla que contiene la columna
  - Los dos privilegios o autorizaciones siguientes:
    - Uno de los siguientes privilegios:
      - Privilegio CONTROL sobre la tabla
      - Privilegio INDEX sobre la tabla
    - Uno de los siguientes privilegios o autorizaciones en el esquema:
      - Autorización IMPLICIT_SCHEMA sobre la base de datos, si el esquema implícito o explícito del índice no existe
      - Privilegio CREATEIN sobre el esquema, si el nombre de esquema del índice hace referencia a un esquema existente
- Debe conocer los valores que desea especificar para el nombre calificado al completo del índice reticular espacial y los tres tamaños de retícula que utilizará el índice. Para obtener más información, consulte el apartado "Determinación de los tamaños de retícula para un índice reticular espacial" en la página 116.

Recomendaciones:

- Para crear un índice reticular espacial en una columna, utilice Index Advisor para determinar los parámetros del índice. Index Advisor puede analizar datos de columnas espaciales y sugerir tamaños de retícula apropiados para el índice reticular espacial utilizado.
- Si prevé realizar una carga inicial de datos en la columna, debería crear el índice reticular espacial después de finalizar el proceso de carga. De esta forma, puede seleccionar los tamaños óptimos de las celdas reticulares de acuerdo con las características de los datos, utilizando Index Advisor. Además, el rendimiento del proceso de carga será mayor si carga los datos antes de crear el índice, pues no será necesario realizar el mantenimiento del índice reticular espacial durante el proceso de carga.

Restricciones:

Las mismas restricciones que existen al crear índices mediante la sentencia CREATE INDEX son aplicables cuando crea un índice reticular espacial . Es decir, la columna en la que crea un índice debe ser una columna de tabla base, no una columna de vista ni una columna de seudónimo. DB2 UDB resolverá los alias durante el proceso.

Procedimiento:

Puede crear un índice reticular espacial de cualquiera de las maneras siguientes:

- Utilice la ventana Spatial Extender del Centro de control de DB2.
- Utilice la sentencia CREATE INDEX de SQL con la extensión db2gse.spatial_index en la cláusula EXTEND USING.
- Utilice una herramienta GIS que funcione con DB2 Spatial Extender. Si utiliza dicha herramienta para crear el índice, la herramienta emitirá la sentencia CREATE INDEX de SQL apropiada.

Esta sección muestra los pasos para los dos primeros métodos. Para obtener información sobre la utilización de una herramienta GIS para crear un índice reticular espacial, consulte la documentación proporcionada con dicha herramienta.

Para crear un índice reticular espacial utilizando el Centro de control:

1. En el Centro de control, pulse con el botón derecho sobre la tabla que tiene la columna espacial para la que desea crear un índice reticular espacial y seleccione **Spatial Extender→Índices espaciales** en el menú emergente. Se abrirá la ventana Índices espaciales.
2. Siga las instrucciones de la ayuda en línea para la ventana Índices espaciales. Para visualizar dichas instrucciones, pulse el botón **Ayuda** en la ventana Índices espaciales.

Para realizar esta tarea utilizando la sentencia CREATE INDEX de SQL:

1. Determine la sentencia CREATE INDEX utilizando la cláusula EXTEND USING y la extensión de índice reticular db2gse.spatial_index.

   Por ejemplo, la siguiente sentencia crea el índice reticular espacial TERRIDX para la tabla BRANCHES que tiene una columna espacial TERRITORY.

   ```
   CREATE INDEX terridx
      ON branches (territory)
      EXTEND USING db2gse.spatial_index (1.0, 10.0, 100.0)
   ```
2. Emita el mandato CREATE INDEX desde el Editor de mandatos de DB2, la Ventana de mandatos de DB2 o el procesador de línea de mandatos de DB2.

Conceptos relacionados:

- "Índices reticulares espaciales" en la página 104
- "Consideraciones sobre el número de niveles de índice y tamaños de retícula" en la página 106
- "Ajuste de índices reticulares espaciales con Index Advisor—Visión general" en la página 115

Tareas relacionadas:

- "Determinación de los tamaños de retícula para un índice reticular espacial" en la página 116
- "Análisis de estadísticas de índices reticulares espaciales" en la página 117

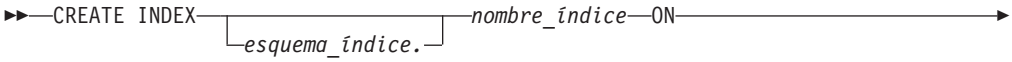
Información relacionada:

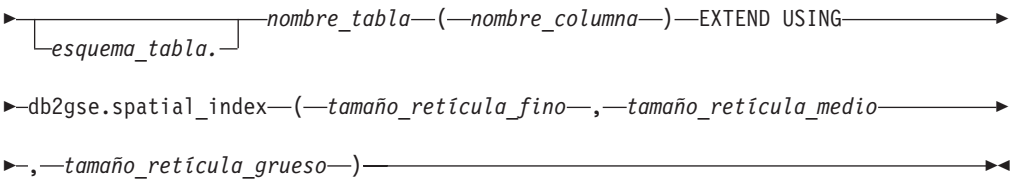
- "Sentencia CREATE INDEX" en el manual *Consulta de SQL, Volumen 2*
- "Sentencia CREATE INDEX para un índice reticular espacial" en la página 113
- "Funciones que utilizan índices para optimizar consultas" en la página 128

Sentencia CREATE INDEX para un índice reticular espacial

Utilice la sentencia CREATE INDEX con la cláusula EXTEND USING para crear un índice reticular espacial.

Sintaxis:





Parámetros:

esquema_índice.
Nombre del esquema al que debe pertenecer el índice que está creando. Si no especifica un nombre, DB2 UDB utiliza el nombre de esquema almacenado en el registro especial CURRENT SCHEMA.

nombre_índice
Nombre, no calificado, del índice reticular que está creando.

esquema_tabla.
Nombre del esquema al que pertenece la tabla donde reside *nombre_columna*. Si no especifica un nombre, DB2 utiliza el nombre de esquema que está almacenado en el registro especial CURRENT SCHEMA.

nombre_tabla
Nombre, no calificado, de la tabla donde reside *nombre_columna*.

nombre_columna
Nombre de la columna espacial para la que se crea el índice reticular espacial.

tamaño_retícula_fino, tamaño_retícula_medio, tamaño_retícula_grueso
Tamaños de retícula para el índice reticular espacial. Estos parámetros deben cumplir estas condiciones:

- *tamaño_retícula_fino* debe ser mayor que 0.
- *tamaño_retícula_medio* debe ser mayor que *tamaño_retícula_fino* o igual a 0.
- *tamaño_retícula_grueso* debe ser mayor que *tamaño_retícula_medio* o igual que 0.

Cuando crea el índice reticular espacial utilizando el Centro de control o la sentencia CREATE INDEX, la validez de los tamaños de retícula se comprueba al indexar la primera geometría. Por lo tanto, si los tamaños de retícula que especifica no cumplen las condiciones de sus valores, se emite una condición de error tal como se describe en estas situaciones:

- Si todas las geometrías de la columna espacial son nulas, Spatial Extender crea satisfactoriamente el índice sin verificar la validez de los tamaños de retícula. Spatial Extender valida los tamaños de retícula cuando el usuario inserta o actualiza una geometría que no sea nula en esa columna espacial. Si los tamaños de retícula especificados no son válidos, se produce un error al insertar o actualizar la geometría no nula.
- Si hay geometrías no nulas en la columna espacial cuando el usuario crea el índice, Spatial Extender valida los tamaños de retícula en ese momento. Si los tamaños de retícula especificados no son válidos, se produce un error inmediatamente y no se crea el índice reticular espacial.

Ejemplos:

En el siguiente ejemplo, la sentencia CREATE INDEX crea el índice reticular espacial TERRIDX en la columna espacial TERRITORY de la tabla BRANCHES:

```
CREATE INDEX terridx
   ON branches (territory)
   EXTEND USING db2gse.spatial_index (1.0, 10.0, 100.0)
```

Conceptos relacionados:

- "Consideraciones sobre el número de niveles de índice y tamaños de retícula" en la página 106
- "Índices reticulares espaciales" en la página 104
- "Ajuste de índices reticulares espaciales con Index Advisor—Visión general" en la página 115

Tareas relacionadas:

- "Determinación de los tamaños de retícula para un índice reticular espacial" en la página 116
- "Análisis de estadísticas de índices reticulares espaciales" en la página 117
- "Creación de índices reticulares espaciales" en la página 111

Información relacionada:

- "Funciones que utilizan índices para optimizar consultas" en la página 128

Ajuste de índices reticulares espaciales con Index Advisor

Ajuste de índices reticulares espaciales con Index Advisor—Visión general

DB2® Spatial Extender proporciona un programa de utilidad, llamado *Index Advisor*, que puede utilizar para:

- Determinar los tamaños de retícula apropiados para sus índices reticulares espaciales.

  Index Advisor analiza las geometrías de una columna espacial y recomienda tamaños de retícula óptimos para el índice reticular espacial. Para conocer el procedimiento, consulte el apartado "Determinación de los tamaños de retícula para un índice reticular espacial" en la página 116.
- Analizar un índice reticular existente.

  Index Advisor puede recopilar y visualizar estadísticas a partir de las que puede determinar cómo facilitarán la recuperación de los datos espaciales los tamaños de celda reticular actuales. Para conocer el procedimiento, consulte el apartado "Análisis de estadísticas de índices reticulares espaciales" en la página 117.

Conceptos relacionados:

- "Consideraciones sobre el número de niveles de índice y tamaños de retícula" en la página 106
- "Índices reticulares espaciales" en la página 104

Tareas relacionadas:

- "Determinación de los tamaños de retícula para un índice reticular espacial" en la página 116
- "Análisis de estadísticas de índices reticulares espaciales" en la página 117

Información relacionada:

- "Sentencia CREATE INDEX para un índice reticular espacial" en la página 113

- "El mandato gseidx" en la página 122
- "Funciones que utilizan índices para optimizar consultas" en la página 128

Determinación de los tamaños de retícula para un índice reticular espacial

Para crear un índice reticular espacial en una columna, utilice Index Advisor para determinar los tamaños de retícula apropiados.

Requisitos previos:

Para poder analizar los datos que desea indexar:

- Su ID de usuario debe tener privilegio SELECT sobre esta tabla.
- Si la tabla tiene más de un millón de filas, probablemente deseará utilizar la cláusula ANALYZE para analizar un subconjunto de las filas y disponer de un tiempo de proceso razonable. Deberá tener disponible un espacio de tablas USER TEMPORARY para utilizar la cláusula ANALYZE. Establezca el tamaño de página de este espacio de tablas en 8 KB como mínimo y asegúrese de disponer de privilegios USE sobre el mismo. Por ejemplo, las siguientes sentencias de DDL crean una agrupación de almacenamientos intermedios con el mismo tamaño de página que el espacio de tablas temporal y otorgan el privilegio USE a cualquiera:

  ```
  CREATE BUFFERPOOL bp8k  SIZE 1000 PAGESIZE 8 K;
  CREATE USER TEMPORARY TABLESPACE usertempts
     PAGESIZE 8K
     MANAGED BY SYSTEM USING ('c:\tempts')
     BUFFERPOOL bp8k
  GRANT USE OF TABLESPACE usertempts TO PUBLIC;
  ```

  Otra alternativa consiste en utilizar el Centro de control de DB2 para crear un espacio en la tabla del usuario y la agrupación de almacenamientos intermedios correspondiente.

Procedimiento:

Para determinar los tamaños de retícula apropiados para un índice reticular espacial:

1. Solicite a Index Advisor que recomiende los tamaños de las celdas de la retícula para el índice que desea crear.
   a. Entre el mandato que invoca al Index Advisor con la palabra clave ADVISE para solicitar los tamaños de las celdas de la retícula. Por ejemplo, para invocar al Index Advisor para la columna SHAPE de la tabla COUNTIES, escriba:

      ```
      gseidx CONNECT TO mi_bd USER ID_usuario USING contraseña GET GEOMETRY
      STATISTICS FOR COLUMN ID_usuario.counties(shape) ADVISE
      ```

      Restricción: Si entra el mandato **gseidx** anterior desde una solicitud del sistema operativo, debe escribir todo el mandato en una sola línea. De forma alternativa, puede ejecutar los mandatos **gseidx** desde un archivo CLP, lo que permite dividir el mandato en varias líneas.

      Index Advisor devolverá los tamaños de celda reticular recomendados. Por ejemplo, el mandato **gseidx** anterior con la palabra clave ADVISE devuelve los siguientes tamaños de celda recomendados para la columna SHAPE:

```
Tamaño de la ven-  Tamaños de retícula
tana de consulta   recomendados                    Coste
-----------------  --------------------------      ----
       0.1          0.7,       2.8,      14.0       2.7
       0.2          0.7,       2.8,      14.0       2.9
       0.5          1.4,       3.5,      14.0       3.5
         1          1.4,       3.5,      14.0       4.8
         2          1.4,       3.5,      14.0       8.2
         5          1.4,       3.5,      14.0        24
        10          2.8,       8.4,      21.0        66
        20          4.2,      14.7,      37.0       190
        50          7.0,      14.0,      70.0       900
       100         42.0,         0,         0      2800
```

b. Elija un tamaño apropiado para la ventana de consulta en la salida de **gseidx**, basado en la anchura de las coordenadas que visualiza en la pantalla.

   En este ejemplo, los valores de latitud y longitud en grados decimales representan las coordenadas. Si la pantalla de mapa típica tiene una anchura de unos 0,5 grados (aproximadamente 55 kilómetros), vaya a la fila que tiene el valor de 0,5 en la columna Tamaño de la ventana de consulta. Esta fila tiene unos tamaños de retícula recomendados de 1,4, 3,5 y 14,0.

2. Cree el índice con los tamaños de retícula recomendados. Para el ejemplo del paso anterior puede ejecutar la siguiente sentencia de SQL:

```
CREATE INDEX counties_shape_idx ON userID.counties(shape)
EXTEND USING DB2GSE.SPATIAL_INDEX(1.4,3.5,14.0);
```

Conceptos relacionados:

- "Consideraciones sobre el número de niveles de índice y tamaños de retícula" en la página 106
- "Índices reticulares espaciales" en la página 104

Tareas relacionadas:

- "Análisis de estadísticas de índices reticulares espaciales" en la página 117

Información relacionada:

- "Sentencia CREATE INDEX para un índice reticular espacial" en la página 113
- "El mandato gseidx" en la página 122
- "Funciones que utilizan índices para optimizar consultas" en la página 128

Análisis de estadísticas de índices reticulares espaciales

Las estadísticas obtenidas para un índice reticular espacial existente pueden indicarle si el índice es eficiente o si debería sustituirse por otro más eficiente. Utilice Index Advisor para obtener estas estadísticas y, si es necesario, para sustituir el índice.

Recomendación: Tan importante como ajustar el índice lo es comprobar si se utiliza para las consultas. Para determinar si se está utilizando un índice espacial, ejecute Visual Explain en el Centro de control de DB2 o una herramienta de línea de mandatos como **db2exfmt** para la consulta. En la sección "Plan de acceso" de la salida de explicación, si ve un operador de EISCAN y el nombre del índice espacial significa que la consulta utiliza el índice.

Requisitos previos:

Para poder analizar los datos que desea indexar:

- Su ID de usuario debe tener privilegio SELECT sobre esta tabla.
- Si la tabla tiene más de un millón de filas, probablemente deseará utilizar la cláusula ANALYZE para analizar un subconjunto de las filas y disponer de un tiempo de proceso razonable. Deberá tener disponible un espacio de tablas USER TEMPORARY para utilizar la cláusula ANALYZE. Establezca el tamaño de página de este espacio de tablas en 8 KB como mínimo y asegúrese de disponer de privilegios USE sobre el mismo. Por ejemplo, las siguientes sentencias de DDL crean una agrupación de almacenamientos intermedios con el mismo tamaño de página que el espacio de tablas temporal y otorgan el privilegio USE a cualquiera:

  ```
  CREATE BUFFERPOOL bp8k  SIZE 1000 PAGESIZE 8 K;
  CREATE USER TEMPORARY TABLESPACE usertempts
     PAGESIZE 8K
     MANAGED BY SYSTEM USING ('c:\tempts')
     BUFFERPOOL bp8k
  GRANT USE OF TABLESPACE usertempts TO PUBLIC;
  ```

  Otra alternativa consiste en utilizar el Centro de control de DB2 para crear un espacio en la tabla del usuario y la agrupación de almacenamientos intermedios correspondiente.

Procedimiento:

Para obtener estadísticas en un índice reticular espacial, si es necesario, con el propósito de sustituir el índice:

1. Haga que Index Advisor recopile estadísticas de acuerdo con los tamaños de celda reticular del índice existente. Puede solicitar estadísticas para todos los datos o para un subconjunto de los datos indexados.
   - Para obtener estadísticas de los datos indexados en un subconjunto de filas, entre el mandato **gseidx** y especifique la palabra clave ANALYZE y sus parámetros, además de la cláusula existing-index y de palabra clave DETAIL. Puede especificar el número o el porcentaje de filas que Index Advisor debe analizar para obtener estadísticas. Por ejemplo, para obtener estadísticas sobre un subconjunto de los datos indexados por el índice COUNTIES_SHAPE_IDX, entre:

     ```
     gseidx CONNECT TO mi_bd USER ID_usuario USING contraseña GET GEOMETRY
     STATISTICS FOR INDEX ID_usuario.counties_shape_idx DETAIL ANALYZE 25 PERCENT
     ADVISE
     ```

   - Para obtener estadísticas sobre todos los datos indexados, entre el mandato **gseidx** y especifique su cláusula existing-index. Incluya la palabra clave DETAIL. Por ejemplo, para invocar a Index Advisor para el índice COUNTIES_SHAPE_IDX, entre:

     ```
     gseidx CONNECT TO mi_bd USER ID_usuario USING contraseña GET GEOMETRY
     STATISTICS FOR INDEX ID_usuario.counties_shape_idx
     DETAIL SHOW HISTOGRAM ADVISE
     ```

   Index Advisor devuelve estadísticas, un histograma de los datos y tamaños de celda recomendados para el índice existente. Por ejemplo, el mandato anterior **gseidx** para todos los datos indexados por COUNTIES_SHAPE_IDX devuelve las siguientes estadísticas:

   ```
   Nivel reticular 1
   -----------------

   Tamaño de retícula                    : 0.5
   ```

```
Número de geometrías                   : 2936
Número de entradas de índice           : 12197

Número de celdas reticulares ocupadas           : 2922
Proporción de entradas de índice/geometría      : 4.154292
Proporción de geometrías/celda reticular        : 1.004791
Número máximo de geometrías por celda reticular : 14
Número mínimo de geometrías por celda reticular : 1

Entradas de índice:  1      2      3      4      10
-------------------  ------ ------ ------ ------ ------
Absoluto          :  86     564    72     1519   695
Porcentaje (%)    :  2.93   19.21  2.45   51.74  23.67



Nivel reticular 2
-----------------

Tamaño reticular                  : 0.0
No hay geometrías indexadas en este nivel.


Nivel reticular 3
-----------------

Tamaño reticular                  : 0.0
No hay geometrías indexadas en este nivel.


Nivel reticular X
-----------------

Número de geometrías          : 205
Número de entradas de índice  : 205
```

2. Determine en qué grado los tamaños de celda reticular del índice existente son apropiados para la obtención de datos. Evalúe las estadísticas devueltas en el paso anterior.

   Consejos:

   - La estadística "Proporción de entradas de índice/geometría" debería ser un valor comprendido entre 1 y 4, preferiblemente valores cercanos a 1.
   - El número de entradas de índice por geometría debería ser menor que 10 en el tamaño reticular más grande para evitar el nivel de desbordamiento.

     La apariencia de la sección "Nivel reticular X" de la salida de Index Advisor indica que existe un nivel de desbordamiento.

   Las estadísticas de índice obtenidas en el paso anterior para COUNTIES_SHAPE_IDX indican que los tamaños de retícula (0.5, 0, 0) no son adecuados para los datos de esta columna porque:

   - Para el Nivel reticular 1, el valor de "Proporción de entradas de índice/geometría" `4.154292` es mayor que la directriz de 4.

     La línea "Entradas de índice" tiene los valores 1, 2, 3, 4 y 10, que indican el número de entradas de índice por geometría. Los valores "Absolutos" que se encuentran debajo de cada columna "Entradas de índice" indican el número de geometrías que tienen aquel número específico de entradas de índice. Por ejemplo, la salida del paso anterior muestra que 1519 geometrías tienen 4 entradas de índice. El valor "Absoluto" de 10 entradas de índice es de 695, lo que indica que 695 geometrías tienen entre 5 y 10 entradas de índice.

- La apariencia de la sección “Nivel reticular X” indica que existe un nivel de índice de desbordamiento. Las estadísticas muestran que 205 geometrías tienen más de 10 entradas de índice cada una.

3. Si las estadísticas no son satisfactorias, observe la sección “Histograma” y las filas apropiadas en las columnas “Tamaño de la ventana de consulta” y “Tamaños de retícula recomendados” en la salida de Index Advisor:
   a. Averigüe el tamaño de MBR que tiene el número más grande de geometrías. La sección “Histograma” lista los tamaños de MBR y el número de geometrías que tienen dichos tamaños de MBR. En el siguiente histograma de ejemplo, el número más grande geometrías (437) está en el tamaño de MBR 0.5.

```
Histograma:
----------
    Tamaño de MBR        Número de geometrías
    -------------------- --------------------
                0.040000                    1
                0.045000                    3
                0.050000                    1
                0.055000                    3
                0.060000                    3
                0.070000                    4
                0.075000                    3
                0.080000                    4
                0.085000                    1
                0.090000                    2
                0.095000                    1
                0.150000                   10
                0.200000                    9
                0.250000                   15
                0.300000                   23
                0.350000                   83
                0.400000                  156
                0.450000                  282
                0.500000                  437
                0.550000                  397
                0.600000                  341
                0.650000                  246
                0.700000                  201
                0.750000                  154
                0.800000                  120
                0.850000                   66
                0.900000                   79
                0.950000                   59
                1.000000                   47
                1.500000                  230
                2.000000                   89
                2.500000                   34
                3.000000                   10
                3.500000                    5
                4.000000                    3
                5.000000                    3
                5.500000                    2
                6.000000                    2
                6.500000                    3
                7.000000                    2
                8.000000                    1
               15.000000                    3
               25.000000                    2
               30.000000                    1
```

   b. Vaya a la fila de Tamaño de la ventana de consulta que tenga el valor de 0.5 para obtener los tamaños de retícula recomendados (1.4, 3.5, 14.0).

```
Tamaño de la ven-  Tamaños de retícula
tana de consulta   recomendados                    Coste
-----------------  --------------------------      ----
       0.1          0.7,       2.8,      14.0       2.7
       0.2          0.7,       2.8,      14.0       2.9
       0.5          1.4,       3.5,      14.0       3.5
         1          1.4,       3.5,      14.0       4.8
         2          1.4,       3.5,      14.0       8.2
         5          1.4,       3.5,      14.0        24
        10          2.8,       8.4,      21.0        66
        20          4.2,      14.7,      37.0       190
        50          7.0,      14.0,      70.0       900
       100         42.0,         0,         0      2800
```

4. Compruebe si los tamaños recomendados se ajustan a las directrices del paso 2. Ejecute el mandato **gseidx** con los tamaños de retícula recomendados:

```
gseidx CONNECT TO mi_bd USER ID_usuario USING contraseña GET GEOMETRY
 STATISTICS FOR COLUMN ID_usuario.counties(shape)
 USING GRID SIZES (1.4, 3.5, 14.0)

Nivel reticular 1
-----------------

Tamaño de retícula            : 1.4
Número de geometrías          : 3065
Número de entradas de índice  : 5951

Número de celdas reticulares ocupadas           : 513
Proporción de entradas de índice/geometría      : 1.941599
Proporción de geometrías/celda reticular        : 5.974659
Número máximo de geometrías por celda reticular : 42
Número mínimo de geometrías por celda reticular : 1

Entradas de índice:  1      2      3      4      10
-------------------  ------ ------ ------ ------ ------
Absoluto          :  1180   1377   15     493    0
Porcentaje (%)    :  38.50  44.93  0.49   16.08  0.00


Nivel reticular 2
-----------------

Tamaño reticular              : 3.5
Número de geometrías          : 61
Número de entradas de índice  : 143

Número de celdas reticulares ocupadas           : 56
Proporción de entradas de índice/geometría      : 2.344262
Proporción de geometrías/celda reticular        : 1.089286
Número máximo de geometrías por celda reticular : 10
Número mínimo de geometrías por celda reticular : 1

Entradas de índice:  1      2      3      4      10
-------------------  ------ ------ ------ ------ ------
Absoluto          :  15     28     0      18     0
Porcentaje (%)    :  24.59  45.90  0.00   29.51  0.00


Nivel reticular 3
-----------------

Tamaño de retícula            : 14.0
Número de geometría           : 15
Número de entradas de índice  : 28

Número de celdas reticulares ocupadas           : 9
Proporción de entradas de índice/geometría      : 1.866667
Proporción de geometrías/celda reticular        : 1.666667
```

```
Número máximo de geometrías por celda reticular : 10
Número mínimo de geometrías por celda reticular : 1

Entradas de índice:  1      2      3      4      10
-------------------  ------ ------ ------ ------ ------
Absoluto          :  7      5      1      2      0
Porcentaje (%)    :  46.67  33.33  6.67   13.33  0.00
```

Ahora las estadísticas muestran los valores contenidos en las directrices:

- Los valores de "Proporción de entradas de índice/geometría" son de 1.941599 para el Nivel reticular 1, de 2.344262 para el Nivel reticular 2 y de 1.866667 para el Nivel reticular 3. Todos estos valores están contenidos en el valor de rango de 1 a 4 de las directrices.
- La ausencia de la sección "Nivel reticular X" indica que ninguna entrada de índice se encuentra en el nivel de desbordamiento.

5. Descarte el índice existente y sustitúyalo por un índice que especifique los tamaños de retícula aconsejados. Para el ejemplo del paso anterior, ejecute las siguientes sentencias de DDL:

```
DROP INDEX ID_usuario.counties_shape_idx;
CREATE INDEX counties_shape_idx ON ID_usuario.counties(shape) EXTEND USING
  DB2GSE.SPATIAL_INDEX(1.4,3.5,14.0);
```

Conceptos relacionados:

- "Consideraciones sobre el número de niveles de índice y tamaños de retícula" en la página 106
- "Índices reticulares espaciales" en la página 104

Tareas relacionadas:

- "Determinación de los tamaños de retícula para un índice reticular espacial" en la página 116
- "Creación de índices reticulares espaciales" en la página 111

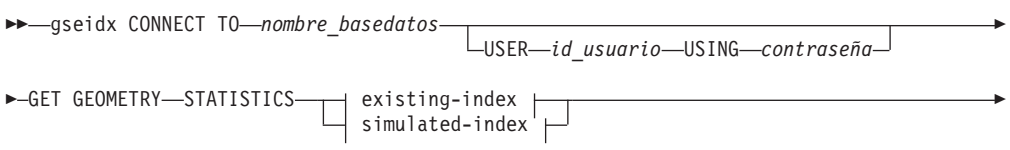
Información relacionada:

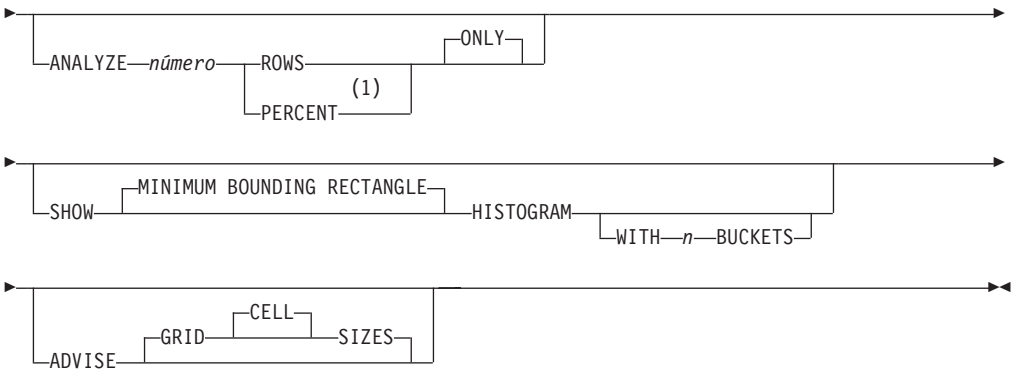
- "Sentencia CREATE INDEX para un índice reticular espacial" en la página 113
- "El mandato gseidx" en la página 122
- "Funciones que utilizan índices para optimizar consultas" en la página 128

El mandato gseidx

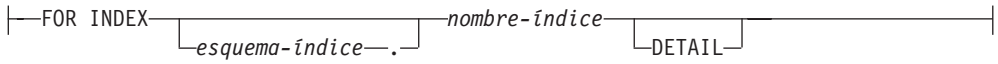
Utilice el mandato **gseidx** para invocar a Index Advisor y manejar índices reticulares espaciales.

Sintaxis

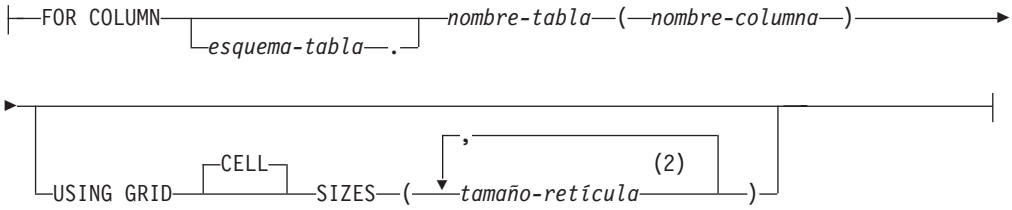




existing-index:



simulated-index:



Notas:

1 En lugar de la palabra clave PERCENT, puede especificar un símbolo de porcentaje (%).

2 Puede especificar tamaños de celda para uno, dos o tres niveles reticulares.

Parámetros:

nombre_basedatos
El nombre de la base de datos en la que reside la tabla espacial.

id_usuario
El ID de usuario que tiene autorización SYSADM o DBADM sobre la base de datos en la que reside el índice o la tabla o autorización SELECT sobre la tabla. Si inicia la sesión en el entorno de mandatos de DB2 con el ID de usuario del propietario de la base de datos no tendrá que especificar *ID_usuario* ni *contraseña* con el mandato **gseidx**.

contraseña
La contraseña para el ID de usuario.

existing-index:

Especifica un índice existente de acuerdo con el cual se recogen estadísticas.

esquema-índice
Nombre del esquema donde está contenido el índice existente.

nombre-índice
Nombre no calificado del índice existente.

DETAIL
Muestra la información siguiente sobre cada nivel reticular:
- El tamaño de las celdas reticulares
- El número de geometrías indexadas
- El número de entradas de índice
- El número de celdas reticulares que contienen geometrías
- El número promedio de entradas de índice por geometría
- El número promedio de geometrías por celda reticular
- El número de geometrías de la celda que contiene el mayor número de geometrías
- El número de geometrías de la celda que contiene el menor número de geometrías

simulated-index:

Especifica una columna de tabla y un índice simulado para esta columna.

esquema-tabla
Nombre del esquema donde reside la tabla con la columna para la que está pensado el índice simulado.

nombre-tabla
Nombre no calificado de la tabla con la columna para la que está pensado el índice simulado.

nombre-columna
Nombre no calificado de la columna de tabla para la que está pensado el índice simulado.

tamaño-retícula
Tamaño de las celdas de cada nivel reticular (fino, medio y grueso) de un índice simulado. Debe especificar un tamaño de celda para un nivel como mínimo. Si no desea incluir un nivel, no especifique un tamaño de celda reticular para él o especifique el valor 0.0 (cero) como tamaño de la celda reticular del nivel.

Cuando se especifica el parámetro *tamaño-retícula*, Index Advisor proporciona la misma clase de estadísticas que cuando se especifica la palabra clave DETAIL en la cláusula existing-index.

ANALYZE *número* **ROWS | PERCENT ONLY**
Reúne estadísticas sobre los datos de un subconjunto de filas de tabla. Si la tabla tiene más de un millón de filas, probablemente deseará utilizar la cláusula ANALYZE para disponer de suficiente tiempo de proceso. Especifique el número o porcentaje aproximado de filas que se deben incluir en ese subconjunto.

SHOW MINIMUM BOUNDING RECTANGLE HISTOGRAM
Visualiza un diagrama que muestras los tamaños de los rectángulos delimitadores mínimos (los MBR) y el número de geometrías cuyos MBR son del mismo tamaño.

WITH *n* **BUCKETS**
Especifica el número de agrupaciones de los MBR de todas las geometrías analizadas. Los MBR pequeños se agrupan junto con otras geometrías pequeñas. Los MBR mayores se agrupan junto con otras geometrías mayores.

Si no especifica este parámetro o especifica 0 cubetas, Index Advisor muestra tamaños de cubeta logarítmicos. Por ejemplo, los tamaños de los MBR podrían ser valores logarítmicos tales como 1.0, 2.0, 3.0,... 10.0, 20.0, 30.0,... 100.0, 200.0, 300.0,...

Si especifica un número de cubetas mayor que 0, Index Advisor muestra valores con el mismo tamaño. Por ejemplo, los valores de los MBR podrían ser valores con el mismo tamaño tales como 8.0, 16.0, 24.0,... 320.0, 328.0, 334.0.

Por omisión se utilizan cubetas de tamaño logarítmico.

ADVISE GRID CELL SIZES
Calcula tamaños de celda reticular próximos al valor óptimo.

Nota acerca del uso:

Si entra el mandato **gseidx** desde un indicador del sistema operativo, debe escribir todo el mandato en una sola línea.

Ejemplo:

El ejemplo siguiente solicita información detallada sobre un índice reticular existente cuyo nombre es COUNTIES_SHAPE_IDX y sugiere tamaños de índice reticular apropiados:

```
gseidx CONNECT TO mi_bd USER ID_usuario USING contraseña GET GEOMETRY
STATISTICS FOR INDEX ID_usuario.counties_shape_idx DETAIL ADVISE
```

Para obtener una explicación acerca de la información que devuelve este mandato, consulte el apartado "Análisis de estadísticas de índices reticulares espaciales" en la página 117.

Conceptos relacionados:

- "Consideraciones sobre el número de niveles de índice y tamaños de retícula" en la página 106
- "Índices reticulares espaciales" en la página 104
- "Ajuste de índices reticulares espaciales con Index Advisor—Visión general" en la página 115

Tareas relacionadas:

- "Determinación de los tamaños de retícula para un índice reticular espacial" en la página 116
- "Creación de índices reticulares espaciales" en la página 111

Información relacionada:

- "Sentencia CREATE INDEX para un índice reticular espacial" en la página 113
- "Funciones que utilizan índices para optimizar consultas" en la página 128

Utilización de vistas para acceder a columnas espaciales

Puede definir una vista que hace uso de una columna espacial de la misma forma que define vistas en DB2 para otros tipos de datos. Si tiene una tabla que tiene una columna espacial y desea que sea utilizada por una vista, utilice las fuentes de información siguientes.

Tareas relacionadas:

- "Creating a view" en el manual *Administration Guide: Implementation*

Información relacionada:

- "Sentencia CREATE VIEW" en el manual *Consulta de SQL, Volumen 2*

Capítulo 12. Generación y análisis de información espacial

Después de llenar con datos las columnas espaciales, está preparado para consultarlas. Este capítulo:

- Describe los entornos en los que puede someter consultas
- Proporciona ejemplos de los diversos tipos de funciones espaciales que puede invocar en una consulta
- Proporciona directrices sobre el uso de funciones espaciales en combinación con índices espaciales

Entornos para realizar análisis espaciales

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

- Sentencias de SQL interactivo.

  Puede entrar sentencias de SQL interactivo desde el Editor de mandatos de DB2®, la ventana de mandatos de DB2 o el procesador de línea de mandatos de DB2.
- Programas de aplicación en todos los lenguajes soportados por DB2.

Ejemplos del funcionamiento de las funciones espaciales

DB2 Spatial Extender proporciona funciones que realizan diversas operaciones sobre datos espaciales. En términos generales, estas funciones se pueden clasificar según el tipo de operación que realizan. La Tabla 5 lista las diversas clases de funciones y proporciona ejemplos. El texto que sigue a la Tabla 5 muestra la codificación correspondiente a los ejemplos.

Tabla 5. Funciones espaciales y operaciones

Categoría de función	Ejemplo de operación
Devuelve información sobre geometrías determinadas.	Devolver la extensión, en millas cuadradas, del área de ventas de la Tienda 10.
Realiza comparaciones.	Determinar si el lugar de residencia de un cliente queda dentro del área de ventas de la Tienda 10.
Obtiene nuevas geometrías a partir de otras existentes.	Obtener el área de ventas de una tienda a partir de su ubicación.
Convierte geometrías entre formatos de intercambio de datos.	Convertir en una geometría la información sobre el cliente expresada en formato GML, para que la información se pueda añadir a una base de datos DB2.

Ejemplo 1: Devolver información sobre geometrías determinadas:

En este ejemplo, la función ST_Area devuelve un valor numérico que representa el área de ventas de la tienda 10. Esta función devuelve el área utilizando las mismas unidades que el sistema de coordenadas utilizado para definir la ubicación del área.

```
SELECT db2gse.ST_Area(sales_area)
FROM   stores
WHERE  id = 10
```

El ejemplo siguiente realiza la misma operación que el anterior, pero ST_Area se ejecuta como método y devuelve el área utilizando la milla cuadrada como unidad.

```
SELECT saleas_area..ST_Area('STATUTE MILE')
FROM   stores
WHERE  id = 10
```

Ejemplo 2: Realizar comparaciones:

En este ejemplo, la función ST_Within compara las coordenadas de la geometría correspondiente al lugar de residencia de un cliente con las coordenadas de una geometría representativa del área de ventas de la tienda 10. Los datos devueltos por la función indicarán si el lugar de residencia está dentro del área de ventas.

```
SELECT c.first_name, c.last_name, db2gse.ST_Within(c.location, s.sales_area)
FROM   customers as c. stores AS s
WHERE  s.id = 10
```

Ejemplo 3: Obtener nuevas geometrías a partir de otras existentes:

En este ejemplo, la función ST_Buffer obtiene una geometría representativa del área de ventas de una tienda a partir de una geometría representativa de la ubicación de la tienda.

```
UPDATE stores
SET    sales_area = db2gse.ST_Buffer(location, 10, 'KILOMETERS')
WHERE  id = 10
```

El ejemplo siguiente realiza la misma operación que el anterior, pero ST_Buffer se ejecuta como método.

```
UPDATE stores
SET    sales_area = location..ST_Buffer(10, 'KILOMETERS')
WHERE  id = 10
```

Ejemplo 4: Convertir geometrías entre formatos de intercambio de datos.:

En este ejemplo, la información sobre un cliente codificada en formato GML se convierte en una geometría, para que se pueda guardar en una base de datos DB2.

```
INSERT
INTO   c.name,c.phoneNo,c.address
VALUES ( 123, 'Mary Anne', Smith', db2gse.ST_Point('
<gml:Point><gml:coord><gml=X>-130.876</gml:X>
<gml:Y>41.120'</gml:Y></gml:coord></gml:Point>, 1) )
```

Funciones que utilizan índices para optimizar consultas

Un grupo especializado de funciones espaciales, denominadas *funciones de comparación*, puede mejorar el rendimiento de las consultas aprovechando un índice reticular espacial o un índice Voronoi geodésico (ambos denominados *índices espaciales*). Estas funciones comparan dos geometrías entre sí. Si los resultados de la comparación satisfacen determinados criterios, la función devuelve un valor de 1; si los resultados no satisfacen los criterios, la función devuelve un valor de 0. Si la comparación no puede realizarse, la función puede devolver un valor nulo.

Por ejemplo, la función ST_Overlaps compara dos geometrías que tienen la misma dimensión (por ejemplo, dos cadenas lineales o dos polígonos). Si las geometrías se solapan parcialmente, y si el espacio abarcado por el solapamiento tiene la misma dimensión que las geometrías, ST_Overlaps devuelve el valor 1.

La Tabla 6 muestra qué funciones de comparación pueden utilizar un índice reticular espacial y qué funciones pueden utilizar un índice Voronoi geodésico:

Tabla 6. Funciones de comparación que pueden utilizar un índice reticular espacial o un índice Voronoi geodésico

Función de comparación	Puede utilizar un índice reticular espacial	Puede utilizar un índice Voronoi geodésico
EnvelopesIntersect	Sí	Sí
ST_Contains	Sí	Sí
ST_Crosses	Sí	No
ST_Distance	Sí	Sí
ST_EnvIntersects	Sí	Sí
ST_Equals	Sí	No
ST_Intersects	Sí	Sí
ST_MBRIntersects	Sí	Sí
ST_Overlaps	Sí	No
ST_Touches	Sí	No
ST_Within	Sí	Sí

Debido al tiempo y memoria necesarios para ejecutar una función, esta ejecución puede suponer un volumen de proceso considerable. Además, cuanto más complejas sean las geometrías que se están comparando, más compleja será la comparación y más tiempo exigirá. Las funciones especializadas listadas anteriormente se pueden ejecutar más rápidamente si pueden utilizar un a índice espacial para localizar geometrías. Para que tales funciones puedan utilizar un índice espacial, siga las siguientes normas:

- Especifique la función en una cláusula WHERE. Si está especificada en una cláusula SELECT, HAVING o GROUP BY, no se puede utilizar un índice espacial.
- La función debe ser la expresión situada en el lado izquierdo del predicado.
- El operador utilizado en el predicado que compara el resultado de la función con otra expresión debe ser un signo de igualdad, con una excepción: la función ST_Distance debe utilizar el operador menor que.
- La expresión situada a la derecha del predicado debe ser la constante 1 , excepto en aquellos casos en que ST_Distance sea la función situada a la derecha.
- La operación debe implicar una búsqueda en una columna espacial en la que se ha definido un índice reticular espacial.

Por ejemplo:

```
SELECT   c.name, c.address, c.phone
FROM customers AS c, bank_branches AS b
WHERE db2gse.ST_Distance(c.location, b.location) < 10000
  and b.branch_id = 3
```

La Tabla 7 en la página 130 muestra formas correctas e incorrectas de crear consultas espaciales para utilizar un índice espacial.

Tabla 7. Demostración de cómo las funciones espaciales pueden cumplir o vulnerar las normas para utilizar un índice espacial.

Consultas que hacen uso de funciones espaciales	Normas vulneradas

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Contains(s.sales_zone,`
`ST_Point(-121.8,37.3, 1)) = 1`
	En este ejemplo no se infringe ninguna condición.

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Length(s.location) > 10`
	La función espacial ST_Length no compara geometrías y no puede utilizar un índice espacial.

`SELECT *`
`FROM stores AS s`
`WHERE 1=db2gse.ST_Within(s.location,:BayArea)`
	La función debe ser una expresión situada en el lado izquierdo del predicado.

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Contains(s.sales_zone,`
`ST_Point(-121.8,37.3, 1)) <> 0`
	Las comparaciones de igualdad deben utilizar la constante entera 1.

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Contains(ST_Polygon`
`('polygon((10 10, 10 20, 20 20, 20 10, 10 10))', 1),`
`ST_Point(-121.8, 37.3, 1) = 1`
	No existe ningún índice espacial en ninguno de los argumentos de la función, por lo que no se puede utilizar ningún índice.

Conceptos relacionados:

- "Consideraciones sobre el número de niveles de índice y tamaños de retícula" en la página 106
- "Índices Voronoi geodésicos" en la página 183
- "Índices reticulares espaciales" en la página 104
- "Ajuste de índices reticulares espaciales con Index Advisor—Visión general" en la página 115

Tareas relacionadas:

- "Creación de índices Voronoi geodésicos" en la página 187
- "Creación de índices reticulares espaciales" en la página 111

Capítulo 13. Mandatos de DB2 Spatial Extender

Este capítulo explica los mandatos utilizados para configurar DB2 Spatial Extender. También explica cómo utilizar estos mandatos para desarrollar proyectos.

Ejecución de mandatos para configurar DB2 Spatial Extender y desarrollar proyectos

Puede utilizar un procesador de línea de mandatos (CLP), denominado db2se, para configurar Spatial Extender y crear proyectos que hacen uso de datos espaciales. En esta sección se describe cómo utilizar db2se para ejecutar mandatos de DB2 Spatial Extender.

Requisitos previos:

Para emitir mandatos de db2se, debe estar autorizado para hacerlo. Para saber qué autorización es necesaria para un mandato determinado, consulte en la Tabla 8 el apartado donde se describe el procedimiento almacenado correspondiente al mandato. Por ejemplo, el mandato **db2se create_srs** necesita las mismas autorizaciones que el procedimiento almacenado db2.ST_create_srs.

Excepción: El mandato **db2se shape_info** no llama a ningún procedimiento almacenado. En su lugar, muestra información sobre el contenido de los archivos de formas.

Procedimiento:

Entre los mandatos de db2se desde el indicador del sistema operativo.

Para saber más sobre qué submandatos y parámetros puede especificar:

- Escriba `db2se` o `db2se -h` y luego pulse Intro. Se visualiza una lista de submandatos de db2se.
- Escriba `db2se` y un submandato, o bien `db2se` y un submandato seguido de `-h`. Luego pulse Intro. Se visualizará la sintaxis necesaria del submandato. En esta sintaxis:
  - Cada parámetro está precedido por un guión y va seguido por un espacio reservado para un valor de parámetro.
  - Los parámetros que están entre corchetes son opcionales. Los demás parámetros son obligatorios.

Importante: Para su comodidad, la sintaxis del mandato se obtiene interactivamente en el monitor; no necesita consultar la sintaxis en ningún otro lugar.

Para emitir un mandato de db2se, escriba `db2se`. A continuación, escriba un submandato, seguido por los parámetros y los valores de parámetro que el submandato necesite. Finalmente, pulse Intro.

En las versiones anteriores los submandatos de Spatial Extender debían ir precedidos por `gseadm` en lugar de por `db2se`. Los scripts de gseadm que haya

creado en versiones anteriores todavía funcionarán en la versión 8.1 pero IBM recomienda que migre los scripts para que utilicen el procesador de línea de mandatos db2se.

Tenga en cuenta que:

- Puede ser necesario escribir el ID de usuario y la contraseña que dan acceso a la base de datos que ha especificado. Por ejemplo, escriba el ID y la contraseña si desea conectarse a la base de datos como un usuario que no sea el propio. Siempre preceda el ID con el parámetro `userId` y preceda la contraseña con el parámetro `pw`.

  Si no especifica un ID de usuario y una contraseña, se utilizarán por omisión el ID de usuario y la contraseña actuales.
- Por omisión, no hay distinción entre mayúsculas y minúsculas en los valores que especifique. Si desea realizar esa distinción, encierre los valores entre comillas dobles. Por ejemplo, para especificar el nombre de tabla en minúsculas, mitabla, escriba lo siguiente: ″mitabla″

  Nota: puede ser necesario que preceda las comillas para evitar que sean interpretadas por el indicador del sistema (shell); por ejemplo, especifique lo siguiente:

  ```
  \"mitabla\"
  ```

  Si un valor que distingue entre mayúsculas y minúsculas está calificado por otro valor que distingue entre mayúsculas y minúsculas, delimite los dos valores separadamente; por ejemplo:

  ```
  "miesquema"."mitabla"
  ```

  Encierre las cadenas de caracteres entre comillas dobles; por ejemplo:

  ```
  "select * from nuevatabla"
  ```

Cuando se ejecuta el mandato de db2se, se invoca al procedimiento almacenado correspondiente al mandato y se realiza la operación solicitada.

Visión general de los mandatos de db2se:

La tabla siguiente indica qué mandatos de db2se se deben emitir para realizar las tareas que intervienen en la configuración de Spatial Extender y la creación de proyectos que hacen uso de datos espaciales. Esta tabla también proporciona ejemplos de mandatos de db2se y hace referencia a información acerca de autorizaciones y parámetros específicos de los mandatos. A la derecha de la tarea, en la segunda columna, verá un enlace o una referencia a información sobre un procedimiento almacenado. Se llama a este procedimiento almacenado cuando se emite el mandato. La autorización para utilizar el procedimiento almacenado es la misma que la autorización para utilizar el mandato; además, el mandato y el procedimiento almacenado comparten los mismos parámetros. Para saber más sobre la autorización y los significados de los parámetros, consulte la sección identificada por la referencia.

Tabla 8. Mandatos de db2se clasificados por tarea

Tarea	Mandato y ejemplo
Crear un sistema de coordenadas.	**db2se create_cs** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_create_coordsys. El ejemplo siguiente crea un sistema de coordenadas denominado "misistcoord". `db2se create_cs mi_bd -coordsysName \"misistcoord\" -definition GEOCS[\"GCS_NORTH_AMERICAN_1983\", DATUM["D_North_American_1983\", SPHEROID[\"GRS_1980\",6387137,298.257222101]], PRIMEM[\"Greenwich\",0],UNIT["Degree\", 0.01745329251994329955]]`
Crear un sistema de referencia espacial.	**db2se create_srs** Los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_create_srs. No es necesaria ninguna autorización. El ejemplo siguiente crea un sistema de referencia espacial denominado "misrs". `db2se create_srs mi_bd -srsName \"misrs\" -srsID 100 -xScale 10 -coordsysName \"GCS_North_American_1983\"`
Descartar un sistema de referencia espacial.	**db2se drop_srs** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_drop_srsdb2gse.ST_drop_srs. El ejemplo siguiente descarta un sistema de referencia espacial denominado "misrs". `db2se drop_srs mi_bd -srsName \"misrs\"`
Suprimir una definición de sistema de coordenadas.	**db2se drop_cs** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_drop_coordsysdb2gse.ST_drop_coordsys. El ejemplo siguiente descarta un sistema de coordenadas denominado "misistcoord". `db2se drop_cs mi_bd -coordsysName \"misistcoord\"`
Inhabilitar una configuración para geocodificar datos automáticamente.	**db2se disable_autogc** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_disable_autogeocoding. El ejemplo siguiente inhabilita la geocodificación automática para una columna geocodificada denominada MICOLUMNA en la tabla MITABLA. `db2se disable_autogc mi_bd -tableName \"mitabla\" -columnName \"micolumna\"`

Tabla 8. Mandatos de db2se clasificados por tarea (continuación)

Tarea	Mandato y ejemplo
Habilitar una base de datos para operaciones espaciales.	**db2se enable_db** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_enable_dbdb2gse.ST_enable_db. El ejemplo siguiente habilita una base de datos denominada MIBD para operaciones espaciales. `db2se enable_db mi_bd`

Exportar datos a un archivo de transferencia SDE.	**db2se export_sde**
	
	Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.GSE_export_sdedb2gse.GSE_export_sde.
	
	El ejemplo siguiente exporta datos de la tabla MITABLA_SDE, que contiene la columna espacial MICOLUMNAESPACIAL, a un archivo de transferencia SDE denominado miarchivo_sde.
	`db2se export_sde mi_bd -tableName \"mitabla_SDE\" -columnName \"micolumnaEspacial\" -fileName /home/myaccount/miarchivo_sde`
	
	El ejemplo siguiente exporta datos de una tabla denominada TABLAESPACIAL a un archivo SDE denominado sdex, que se creará en el cliente DB2. Los mensajes de error e informativos (por ejemplo, la hora a la que ha empezado y finalizado la exportación y cuántas filas se han exportado) se graban en un archivo denominado sdex.export.log.
	`db2se export_sde mi_bd -client -fileName sdex -selectStatement "SELECT * FROM spatialTable" -messagesFile sdex.export.log`

Exportar datos a archivos de formas.	**db2se export_shape** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_export_shape. El ejemplo siguiente exporta una columna espacial denominada MICOLUMNA y su tabla asociada, MITABLA, a un archivo de formas denominado miarchivo_formas. `db2se export_shape mi_bd -fileName /home/myaccount/miarchivo_formas -selectStatement "select * from mitabla"`

Tabla 8. Mandatos de db2se clasificados por tarea (continuación)

Tarea	Mandato y ejemplo

Importar un archivo de transferencia SDE.	**db2se import_sde**
	
	Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.GSE_import_sdedb2gse.GSE_import_sde.
	
	El ejemplo siguiente importa un archivo de transferencia SDE denominado miarchivo_sde a la tabla MITABLA_SDE, que contiene una columna espacial denominada MICOLUMNAESPACIAL. Se enviará una confirmación cada diez registros.
	`db2se import_sde mi_bd -tableName \"mitabla_sde\" -columnName \"miColumnaespacial\" -fileName /home/myaccount/"miarchviosde" -commitScope 10`
	
	El ejemplo siguiente muestra cómo importar un archivo SDE denominado sdex, que se encuentra en el cliente DB2. En este ejemplo, los datos se importan en una tabla denominada TABLASDE (a un ID con nombre de columna) y se emite una confirmación cada 100 registros. Los errores se graban en un archivo denominado sdex.exceptions.
	`db2se import_sde mi_bd -client -filename sdex -srsId 1234 -tableName sdeTable -idColumn id -commitScope 100 -messagesFile sdex.exceptions`

Importar archivos de formas.	**db2se import_shape** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_import_shape. El mandato siguiente importa un archivo de formas denominado miarchivo a una tabla denominada MYTABLE. Durante la importación, los datos espaciales de miarchivo se insertan en una columna MITABLA denominada MICOLUMNA. `db2se import_shape mi_bd -fileName \"miarchivo\" -srsName NAD83_SRS_1 -tableName \"mitabla\" -spatialColumnName \"micolumna\"`
Registrar un geocodificador.	**db2se register_gc** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_register_geocoder. El ejemplo siguiente registra un geocodificador denominado “migeocodificador”, que es utilizado por una función denominada “miesquema.mifunción”. `db2se register_gc mi_bd -geocoderName \"migeocodificador"\ -functionSchema \"miesquema\" -functionName \"mifunción\" -defaultParameterValues "1, 'string',,cast(null as varchar(50))" -vendor myvendor -description "el geocodificador de mi proveedor devuelve texto convencional"`

Tabla 8. Mandatos de db2se clasificados por tarea (continuación)

Tarea	Mandato y ejemplo
Registrar una columna espacial.	**db2se register_spatial_column** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_register_spatial. El ejemplo siguiente registra una columna espacial denominada MICOLUMNA en la tabla MITABLA, con el sistema de referencia espacial "USA_SRS_1". `db2se register_spatial_column mi_bd -tableName \"mitabla\" -columnName \"micolumna\" -srcName USA_SRS_1`
Eliminar los recursos que habilitan una base de datos para operaciones espaciales.	**db2se disable_db** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_disable_dbdb2gse.ST_disable_db. El ejemplo siguiente elimina los recursos que habilitan la base de datos MIBD para operaciones espaciales. `db2se disable_db mi_bd`
Eliminar una configuración para operaciones de geocodificación.	**db2se remove_gc_setup** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_remove_gc_setup. El ejemplo siguiente elimina una configuración para las operaciones de geocodificación que se aplican a una columna espacial denominada MICOLUMNA de la tabla MITABLA. `db2se remove_geocoding_setup mi_bd -tableName \"mitabla\" -columnName \"micolumna\"`
Ejecutar un geocodificador en la modalidad de proceso por lotes.	**db2se run_gc** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_run_gc. El ejemplo siguiente ejecuta un geocodificador en modalidad de proceso por lotes para llenar una columna denominada MICOLUMNA de una tabla denominada MITABLA. `db2se run_gc mi_bd -tableName \"mitabla\" -columnName \"micolumna\"`
Configurar un geocodificador para que se ejecute automáticamente.	**db2se enable_autogeocoding** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_enable_autogeocoding. El ejemplo siguiente configura una geocodificación automática para una columna denominada MICOLUMNA en la tabla MITABLA `db2se enable_autogeocoding mi_bd -tableName \"mitabla\" -columnName \"micolumna\"`

Tabla 8. Mandatos de db2se clasificados por tarea (continuación)

Tarea	Mandato y ejemplo
Configurar operaciones de geocodificación.	**db2se setup_gc** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_setup_geocoding. El ejemplo siguiente configura operaciones de geocodificación para llenar una columna espacial denominada MICOLUMNA en la tabla MITABLA. `db2se setup_gc mi_bd -tableName \"mitabla\" -columnName \"micolumna\" -geocoderName \"db2se_USA_GEOCODER\" -parameterValues "address,city,state,zip,2,90,70,20,1.1,'meter',4.." -autogeocodingColumns address,city,state,zip commitScope 10`

Mostrar información acerca de un archivo de formas y su contenido.	**db2se shape_info**
	
	Para utilizar este mandato, debe:
	• Tener permiso para leer el archivo al que hace referencia el mandato.
	• Poder conectarse con la base de datos que contiene este archivo (si utiliza el parámetro *–database*, que especifica que el sistema busca en la base de datos nombrada sistemas de coordenadas y sistemas de referencia espacial compatibles).
	
	El ejemplo siguiente muestra información acerca del archivo de formas denominado miarchivo, que está ubicado en el directorio actual.
	`db2se shape_info -fileName miarchivo`
	
	El ejemplo siguiente muestra información acerca de un archivo de formas UNIX de ejemplo denominado offices. El parámetro *–database* busca todos los sistemas de coordenadas y sistemas de referencia espacial compatibles en la base de datos nombrada (en este caso, MIBD).
	`db2se shape_info -fileName ~/sqllib/samples/spatial/data/offices -database miBD`

Tabla 8. Mandatos de db2se clasificados por tarea (continuación)

Tarea	Mandato y ejemplo

Mostrar información acerca de un archivo SDE y su contenido.	**db2se sde_info**
	
	Para utilizar este mandato, debe:
	• Tener permiso para leer el archivo al que hace referencia el mandato.
	• Poder conectarse con la base de datos que contiene este archivo (si utiliza el parámetro *–database*, que especifica que el sistema busca en la base de datos nombrada sistemas de coordenadas y sistemas de referencia espacial compatibles).
	
	El ejemplo siguiente muestra información acerca del archivo SDE denominado archivo_sde, que está ubicado en el directorio actual.
	`db2se sde_info -fileName miarchivo`
	
	El ejemplo siguiente muestra información acerca de un archivo SDE denominado sdex y busca en la base de datos denominada MIBD todos los sistemas de coordenadas y sistemas de referencia espacial compatibles.
	`db2se sde_info -fileName data/sdex -database miBD`

Deshacer el registro de un codificador.	**db2se unregister_gc** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_unregister_geocoder. El ejemplo siguiente cancela el registro de un geocodificador denominado “migeocodificador”. `db2se unregister_gc mi_bd -geocoderName \"migeocodificador\"`
Cancelar el registro de una columna espacial.	**db2se unregister_spatial_column** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_unregister_spatial_column. El ejemplo siguiente cancela el registro de una columna espacial denominada MICOLUMNA en la tabla MITABLA. `db2se unregister_spatial_column mi_bd -tableName \"mitabla\" -columnName \"micolumna\"`
Actualizar una definición de sistema de coordenadas.	**db2se alter_cs** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_alter_coordsysdb2gse.ST_alter_coordsys. El ejemplo siguiente actualiza la definición de un sistema de coordenadas denominado “misistcoord” con un nuevo nombre de organización. `db2se alter_cs mi_bd -coordsysName \"misistcoord\" -organization myNeworganizationb -tableName \"mitabla\"`

Tabla 8. Mandatos de db2se clasificados por tarea (continuación)

Tarea	Mandato y ejemplo
Actualizar una definición de sistema de referencia espacial.	**db2se alter_srs** Las autorizaciones necesarias y los parámetros específicos del mandato son los mismos que los del procedimiento almacenado db2gse.ST_alter_srsdb2gse.ST_alter_srs. El ejemplo siguiente modifica un sistema de referencia espacial denominado "misrs" con un xOffset y una descripción diferentes. `db2se alter_srs mi_bd -srsName \"misrs\" -xOffset 35 -description "Este es mi propio sistema de referencia espacial."`

Mandatos

Capítulo 14. Escritura de aplicaciones y utilización del programa de ejemplo

Este capítulo explica cómo escribir aplicaciones de Spatial Extender.

Cómo escribir aplicaciones para DB2 Spatial Extender

Si piensa escribir programas de aplicación que invoquen a procedimientos almacenados o funciones de DB2 Spatial Extender, lea la siguiente información de tareas y consulta:

Conceptos relacionados:

- "El programa de ejemplo de DB2 Spatial Extender" en la página 143

Tareas relacionadas:

- "Cómo llamar a los procedimientos almacenados de DB2 Spatial Extender desde una aplicación" en la página 142
- "Cómo incluir el archivo de cabecera de DB2 Spatial Extender en aplicaciones espaciales" en la página 141

Cómo incluir el archivo de cabecera de DB2 Spatial Extender en aplicaciones espaciales

DB2 Spatial Extender proporciona un archivo de cabecera que define constantes que se pueden utilizar con las funciones y los procedimientos almacenados de DB2 Spatial Extender.

Recomendación: Si piensa llamar a procedimientos almacenados o funciones de DB2 Spatial Extender desde programas C o C++, incluya este archivo de cabecera en las aplicaciones espaciales.

Procedimiento:

Para asegurarse de que las aplicaciones de DB2 Spatial Extender puedan utilizar las definiciones necesarias en este archivo de cabecera:

1. Incluya el archivo de cabecera de DB2 Spatial Extender en el programa de aplicación. El archivo de cabecera tiene el nombre siguiente:

   `db2gse.h`

   El archivo de cabecera está situado en el directorio *db2path*/include, donde *db2path* es el directorio de instalación donde está instalado DB2 Universal Database.
2. Asegúrese de que la vía de acceso al directorio include esté especificada en el `makefile` con la opción de compilación.

Si está creando aplicaciones Windows 64 bits en un sistema Windows 32 bits, cambie el parámetro DB2_LIBS en el archivo `samples/spatial/makefile.nt` para acomodar aplicaciones de 64 bits. Las cambios necesarios se resaltan a continuación:

```
DB2_LIBS = $(DB2_DIR)\lib\Win64\db2cli.lib $(DB2_DIR)\lib\Win64\db2api.lib
```

Cómo llamar a los procedimientos almacenados de DB2 Spatial Extender desde una aplicación

Los procedimientos almacenados de DB2 Spatial Extender se crean cuando habilita la base de datos para operaciones espaciales. Si piensa escribir programas de aplicación que llamen a cualquiera de los procedimientos almacenados de DB2 Spatial Extender, utilice la sentencia CALL de SQL y especifique el nombre del procedimiento almacenado.

Procedimiento:

Para llamar a los procedimientos almacenados de DB2 Spatial Extender, realice las acciones siguientes:

- Para llamar al procedimiento almacenado ST_enable_db, que habilita una base de datos para operaciones espaciales, especifique el nombre del procedimiento almacenado de la forma siguiente:

  ```
  CALL db2gse!ST_enable_db
  ```

  `db2gse!` en esta llamada representa el nombre de la biblioteca de DB2 Spatial Extender. El procedimiento almacenado ST_enable_db es el único en el que necesita incluir un signo de exclamación en la llamada (es decir, `db2gse!`).
- Para llamar a cualquier otro procedimiento almacenado de DB2 Spatial Extender, especifique el nombre del procedimiento almacenado de la siguiente forma, donde `db2gse` es el nombre del esquema para todos los procedimientos almacenados de DB2 Spatial Extender, y *nombre_procedimiento_almacenado* es el nombre del procedimiento almacenado:

  ```
  CALL db2gse.nombre_procedimiento_almacenado
  ```

  Tenga en cuenta que en la llamada anterior no se incluye ningún signo de exclamación.

  Los procedimientos almacenados de DB2 Spatial Extender se muestran en la tabla siguiente.

Tabla 9.

Procedimiento almacenado	Descripción
GSE_export_sde	Exporta una columna espacial y su tabla asociada a un archivo de transferencia SDE.
GSE_import_sde	Importa un archivo de transferencia SDE a una base de datos.
ST_alter_coordsys	Actualiza un atributo de un sistema de coordenadas en la base de datos.
ST_alter_srs	Actualiza un atributo de un sistema de referencia espacial en la base de datos.
ST_create_coordsys	Crea un sistema de coordenadas en la base de datos.
ST_create_srs	Crea un sistema de referencia espacial en la base de datos.
ST_disable_autogeocoding	Especifica que DB2 Spatial Extender dejará de sincronizar una columna geocodificada con sus columnas de geocodificación asociadas.

Tabla 9. (continuación)

Procedimiento almacenado	Descripción
ST_disable_db	Elimina recursos que permiten a DB2 Spatial Extender almacenar datos espaciales y dar soporte a operaciones que se realizan sobre estos datos.
ST_drop_coordsys	Suprime un sistema de coordenadas de la base de datos.
ST_drop_srs	Suprime un sistema de referencia espacial de la base de datos.
ST_enable_autogeocoding	Especifica que DB2 Spatial Extender sincronizará una columna geocodificada con sus columnas de geocodificación asociadas.
ST_enable_db	Proporciona una base de datos con los recursos que necesita para almacenar datos espaciales y dar soporte a operaciones.
ST_export_shape	Exporta los datos seleccionados en la base de datos a un archivo de formas.
ST_import_shape	Importa un archivo de formas a una base de datos.
ST_register_geocoder	Registra un geocodificador que no sea DB2SE_USA_GEOCODER, que forma parte del producto DB2 Spatial Extender.
ST_register_spatial_column	Registra una columna espacial y asocia un sistema de referencia espacial a la misma.
ST_remove_geocoding_setup	Elimina toda la información de configuración de geocodificación para la columna geocodificada.
ST_run_geocoding	Ejecuta un geocodificador en modalidad de proceso por lotes.
ST_setup_geocoding	Asocia una columna que se va a geocodificar a un geocodificador y configura los valores de los parámetros de configuración correspondientes.
ST_unregister_geocoder	Deshace el registro de un geocodificador que no sea DB2SE_USA_GEOCODER.
ST_unregister_spatial_column	Elimina el registro de una columna espacial.

El programa de ejemplo de DB2 Spatial Extender

El programa de ejemplo de DB2® Spatial Extender,`runGseDemo`, tiene dos finalidades. Puede utilizar el programa de ejemplo para familiarizarse con la programación de aplicaciones de DB2 Spatial Extender, y para verificar la instalación de DB2 Spatial Extender. Para obtener más información sobre cómo verificar la instalación de Spatial Extender, consulte el apartado "Tareas relacionadas" al final de este tema.

- En UNIX®, el programa runGseDemo reside en la vía siguiente:
  `$HOME/sqllib/samples/spatial`

  donde `$INICIO` es el directorio de inicio del propietario de la instancia.

- En Windows®, el programa runGseDemo reside en la vía siguiente:
  `c:\Archivos de programa\IBM\sqllib\samples\spatial`

  donde `c:\Archivos de programa\IBM\sqllib` es el directorio en el que ha instalado DB2 Spatial Extender.

El programa de ejemplo `runGseDemo` de DB2 Spatial Extender hace más fácil la programación de aplicaciones. Mediante este programa de ejemplo, puede habilitar una base de datos para operaciones espaciales y realizará análisis espaciales sobre los datos de esa base de datos. Esta base de datos contendrá tablas con información ficticia sobre clientes y sobre zonas con riesgo de inundación. A partir de esta información puede experimentar con Spatial Extender y determinar qué clientes corren el riesgo de sufrir daños a causa de una inundación.

Mediante el programa de ejemplo, puede:

- Ver los pasos habituales necesarios para crear y mantener una base de datos habilitada para operaciones espaciales.
- Conocer cómo invocar a procedimientos almacenados espaciales desde un programa de aplicación.
- Cortar y pegar código de ejemplo en sus propias aplicaciones.

Utilice el siguiente programa de ejemplo para codificar tareas para DB2 Spatial Extender. Por ejemplo, suponga que escribe una aplicación que utiliza la interfaz de base de datos para invocar a procedimientos almacenados de DB2 Spatial Extender. Desde el programa de ejemplo, puede copiar código para personalizar su aplicación. Si no conoce los pasos de programación para DB2 Spatial Extender, puede ejecutar el programa de ejemplo para ver cada paso con detalle. Para obtener instrucciones sobre la ejecución del programa de ejemplo, consulte el apartado “Tareas relacionadas” al final de este tema.

La tabla siguiente describe cada paso en el programa de ejemplo. En cada paso realizará una acción y, en muchos casos, deshará esa acción. Por ejemplo, en el primer paso, habilitará la base de datos espacial y luego la inhabilitará. De esta forma, se familiarizará con muchos de los procedimientos almacenados de Spatial Extender. Para obtener más información sobre los procedimientos almacenados responsables de cada paso, consulte el apartado “Tareas relacionadas” al final de este tema.

Tabla 10. Pasos del programa de ejemplo de DB2 Spatial Extender

Pasos	Acción y descripción
Habilitar o inhabilitar la base de datos espacial	• Habilitar la base de datos espacial Éste es el primer paso necesario para utilizar DB2 Spatial Extender. Una base de datos que ha sido habilitada para operaciones espaciales tiene un conjunto de tipos espaciales, un conjunto de funciones espaciales, un conjunto de predicados espaciales, nuevos tipos de índice y un conjunto de tablas y vistas espaciales de catálogo. • Inhabilitar la base de datos espacial Este paso se suele ejecutar cuando se han habilitado características espaciales para una base de datos incorrecta o cuando ya no es necesario realizar operaciones espaciales en la base de datos en cuestión. Cuando inhabilita una base de datos espacial, elimina el conjunto de tipos espaciales, el conjunto de funciones espaciales, el conjunto de predicados espaciales, los nuevos tipos de índice y el conjunto de tablas y vistas de catálogo espaciales asociados a esa base de datos. • Habilitar la base de datos espacial Igual que en el caso anterior.
Crear o descartar un sistema de coordenadas	• Crear un sistema de coordenadas llamado NORTH_AMERICAN Este paso crea un nuevo sistema de coordenadas en la base de datos. • Descartar el sistema de coordenadas llamado NORTH_AMERICAN Este paso descarta el sistema de coordenadas NORTH_AMERICAN de la base de datos. • Crear un sistema de coordenadas llamado KY_STATE_PLANE Este paso crea un nuevo sistema de coordenadas, KY_STATE_PLANE, que será utilizado por el sistema de referencia espacial creado en el paso siguiente.
Crear o descartar un sistema de referencia espacial	• Crear un sistema de referencia espacial llamado SRSDEMO1 Este paso define un nuevo sistema de referencia espacial (spatial reference system, SRS) que se utiliza para interpretar las coordenadas. El SRS comprende datos sobre geometrías en un formato que se puede guardar en una columna de una base de datos habilitada para operaciones espaciales. Una vez registrado el SRS en una columna espacial determinada, las coordenadas aplicables a esa columna espacial se pueden guardar en la columna correspondiente de la tabla CUSTOMERS. • Descartar el SRS llamado SRSDEMO1 Este paso se realiza cuando el SRS ya no es necesario en la base de datos. Cuando descarta un SRS, elimina la definición del SRS contenida en la base de datos. • Crear el SRS llamado KY_STATE_SRS

Tabla 10. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Crear y llenar las tablas espaciales	• Crear la tabla CUSTOMERS • Llenar la tabla CUSTOMERS La tabla CUSTOMERS representa los datos comerciales que se han almacenado en la base de datos durante varios años. • Alterar la tabla CUSTOMERS añadiendo la columna LOCATION La sentencia ALTER TABLE añade una nueva columna (LOCATION) de tipo ST_Point. Esta columna se llenará geocodificando las columnas de dirección en un paso posterior. • Crear la tabla OFFICES La tabla OFFICES representa, entre otros datos, la zona de ventas de cada oficina de una compañía de seguros. En un paso posterior, la tabla completa se llenará con los datos de atributos procedentes de una base de datos no perteneciente a DB2. Este paso posterior supone importar datos de atributos a la tabla OFFICES desde un archivo de formas.
Llenar las columnas	• Geocodificar los datos de direcciones para la columna LOCATION de la tabla CUSTOMERS con el geocodificador denominado KY_STATE_GC Este paso realiza una geocodificación espacial por lotes mediante la invocación del programa geocodificador. La geocodificación por lotes se suele realizar cuando es necesario geocodificar o volver a geocodificar una parte significativa de la tabla. • Cargar la tabla OFFICES anteriormente creada desde el archivo de formas utilizando el sistema de referencia espacial KY_STATE_SRS Este paso carga la tabla OFFICES con datos espaciales existentes en forma de un archivo de formas. Debido a que la tabla OFFICES ya existe, el programa de carga añadirá los nuevos registros a una tabla existente. • Crear y cargar la tabla FLOODZONES desde el archivo de formas utilizando el sistema de referencia espacial KY_STATE_SRS Este paso carga la tabla FLOODZONES con datos existentes en forma de un archivo de formas. Debido a que la tabla no existe, el programa de carga creará la tabla antes de cargar los datos. • Crear y cargar la tabla REGIONS desde el archivo de formas utilizando el sistema de referencia espacial KY_STATE_SRS
Registrar o desregistrar el geocodificador	• Registrar el geocodificador llamado SAMPLEGC • Desregistrar el geocodificador llamado SAMPLEGC • Registrar el geocodificador KY_STATE_GC Estos pasos registran y desregistran el geocodificador llamado SAMPLEGC y luego crean un nuevo geocodificador, KY_STATE_GC, para utilizarlo en el programa de ejemplo.

Tabla 10. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Crear índices espaciales	• Crear el índice reticular espacial para la columna LOCATION de la tabla CUSTOMERS • Descartar el índice reticular espacial para la columna LOCATION de la tabla CUSTOMERS • Crear el índice reticular espacial para la columna LOCATION de la tabla CUSTOMERS • Crear el índice reticular espacial para la columna LOCATION de la tabla OFFICES • Crear el índice reticular espacial para la columna LOCATION de la tabla FLOODZONES • Crear el índice reticular espacial para la columna LOCATION de la tabla REGIONS Estos pasos crean el índice reticular espacial para las tablas CUSTOMERS, OFFICES, FLOODZONES y REGIONS.
Habilitar la geocodificación automática.	• Configurar la geocodificación para la columna LOCATION de la tabla CUSTOMERS con el geocodificador KY_STATE_GC Este paso asocia la columna LOCATION de la tabla CUSTOMERS con el geocodificador KY_STATE_GC y configura los correspondientes valores de parámetros de geocodificación. • Habilitar la geocodificación automática para la columna LOCATION de la tabla CUSTOMERS Este paso activa la invocación automática del geocodificador. La geocodificación automática hace que las columnas LOCATION, STREET, CITY, STATE y ZIP de la tabla CUSTOMERS se sincronicen entre sí para operaciones subsiguientes de inserción y actualización.
Realizar operaciones de inserción, actualización y supresión sobre la tabla CUSTOMERS	• Insertar algunos registros con una calle diferente • Actualizar algunos registros con una nueva dirección • Suprimir todos los registros de la tabla Estos pasos muestran operaciones de inserción, actualización y supresión sobre las columnas STREET, CITY, STATE y ZIP de la tabla CUSTOMERS. Una vez habilitada la geocodificación automática, los datos que se insertan o actualizan en estas columnas se geocodifican automáticamente en la columna LOCATION. Este proceso se habilitó en el paso anterior.
Inhabilitar la geocodificación automática	• Inhabilitar la geocodificación automática para la columna LOCATION de la tabla CUSTOMERS • Eliminar la definición de geocodificación para la columna LOCATION de la tabla CUSTOMERS • Descartar el índice espacial para la columna LOCATION de la tabla CUSTOMERS Estos pasos inhabilitan la invocación automática del geocodificador y el índice espacial como preparación para el paso siguiente. El paso siguiente vuelve a geocodificar la tabla CUSTOMERS completa. **Recomendación:** si carga un volumen grande de datos espaciales, descarte el índice espacial antes de cargar los datos, y luego vuelva a crearlo una vez cargados los datos.

Tabla 10. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Volver a geocodificar la tabla CUSTOMERS	• Volver a geocodificar la columna LOCATION de la tabla CUSTOMERS con un nivel de precisión menor: 90% en lugar de 100% • Volver a crear el índice espacial para la columna LOCATION de la tabla CUSTOMERS • Habilitar de nuevo la geocodificación automática con un nivel de precisión menor: 90% en lugar de 100% Estos pasos ejecutan el geocodificador en la modalidad de proceso por lotes, crean de nuevo el índice espacial y vuelven a habilitar la geocodificación automática con un nuevo nivel de precisión. Esta acción es recomendable cuando se observa un tasa de errores elevada en el proceso de geocodificación. Si el nivel de precisión está establecido en 100%, la geocodificación de una dirección puede fallar debido a que no puede encontrar una dirección coincidente en los datos de referencia. Mediante la disminución del nivel de precisión, el geocodificador puede tener más probabilidad de encontrar datos coincidentes. Después de volver a geocodificar la tabla en la modalidad de proceso por lotes, se habilita de nuevo la geocodificación automática y se vuelve a crear el índice espacial. Esto facilita el mantenimiento incremental del índice espacial y de la columna espacial para operaciones posteriores de inserción y actualización.
Crear una vista y registrar la columna espacial en la vista	• Crear una vista denominada HIGHRISKCUSTOMERS basada en la unión de las tablas CUSTOMERS y FLOODZONES • Registrar la columna espacial de la vista Estos pasos crean una vista y registran su columna espacial.

Tabla 10. Pasos del programa de ejemplo de DB2 Spatial Extender (continuación)

Pasos	Acción y descripción
Realizar el análisis espacial	• Encontrar el número de clientes atendidos en cada región (ST_Within) • Para las oficinas y clientes de la misma región, encontrar el número de clientes que están dentro de una distancia determinada respecto a cada oficina (ST_Within, ST_Distance) • Para cada región, encontrar los ingresos medios y la prima de cada cliente (ST_Within) • Encontrar el número de zonas con riesgo de inundación que están dentro de cada zona de oficinas (ST_Overlaps) • Encontrar la oficina más cercana respecto al lugar de residencia de un cliente determinado, suponiendo que la oficina esté situada en el centro geométrico de la zona de oficinas (ST_Distance) • Encontrar los clientes cuya ubicación está cerca del límite de una zona determinada con riesgo de inundación (ST_Buffer, ST_Intersects) • Encontrar los clientes expuestos a un riesgo alto que están dentro de una distancia especificada respecto a una oficina determinada (ST_Within) Todos estos pasos hacen uso del procedimiento almacenado sqlRunSpatialQueries. Estos pasos realizan el análisis espacial utilizando predicados y funciones espaciales del SQL de DB2. El optimizador de consultas de DB2 saca provecho del índice espacial definido sobre las columnas espaciales para mejorar el rendimiento de la consulta cuando sea posible.
Exportar datos espaciales a archivos de forma	• Exportar la vista HIGHRISKCUSTOMERS a archivos de forma Este paso muestra un ejemplo de exportación de la vista HIGHRISKCUSTOMERS a archivos de forma. La exportación de datos desde un formato de base de datos a otro formato de archivo permite que la información pueda ser utilizada por otras herramientas (tales como ArcExplorer for DB2). Este paso forma parte del programa runGseDemo.c, pero está inhabilitado con fines de consulta solamente. Puede modificar el programa de ejemplo para especificar la ubicación del archivo de formas de exportación y ejecutar de nuevo el programa de ejemplo.
Exportar e importar archivos de SDE	• Exportar la tabla CUSTOMERS a un archivo de transferencia de SDE • Importar datos desde el archivo de transferencia de SDE recién exportado Estos pasos muestran ejemplos de exportación e importación de archivos de transferencia de SDE. Estos pasos se incluyen en el programa runGseDemo.c pero están como comentario con fines de consulta solamente. Puede modificar el programa de ejemplo para especificar la ubicación del archivo SDE de exportación y ejecutar de nuevo el programa de ejemplo.

Tareas relacionadas:

- "Verificación de la instalación de Spatial Extender" en la página 39
- "Resolución de problemas de instalación" en la página 41
- "Cómo escribir aplicaciones para DB2 Spatial Extender" en la página 141
- "Cómo llamar a los procedimientos almacenados de DB2 Spatial Extender desde una aplicación" en la página 142
- "Cómo incluir el archivo de cabecera de DB2 Spatial Extender en aplicaciones espaciales" en la página 141

Capítulo 15. Identificación de problemas de DB2 Spatial Extender

Si tiene un problema al trabajar con DB2 Spatial Extender, necesita determinar la causa del problema. Puede determinar la causa de los problemas de DB2 Spatial Extender de estas maneras:

- Puede utilizar la información de los mensajes para diagnosticar el problema.
- Cuando se utilizan procedimiento almacenados y funciones de Spatial Extender, DB2 proporciona información sobre la ejecución satisfactoria o errónea del procedimiento almacenado o función. La información devuelta consta de un código de mensaje (en forma de número entero), el texto del mensaje, o ambas cosas, dependiendo de la interfaz que utilice para trabajar con DB2 Spatial Extender.
- Puede examinar el archivo de notificación de administración de DB2, en el cual se registra información de diagnóstico sobre errores.
- Si tiene un problema recurrente de Spatial Extender que se pueda reproducir, el soporte técnico de IBM le puede solicitar que utilice el programa de rastreo de DB2 para facilitar el diagnóstico del problema.

Este capítulo trata estos temas.

Cómo interpretar los mensajes de DB2 Spatial Extender

Puede trabajar con DB2® Spatial Extender a través de cuatro interfaces diferentes:

- Procedimientos almacenados de DB2 Spatial Extender
- Funciones de DB2 Spatial Extender
- Procesador de Línea de Mandatos (CLP) de DB2 Spatial Extender
- Centro de control de DB2

Todas las interfaces emiten mensajes de DB2 Spatial Extender para ayudar al usuario a determinar si la operación espacial solicitada se ejecutó satisfactoriamente o produjo un error.

La siguiente tabla describe cada parte del texto de este mensaje de ejemplo de DB2 Spatial Extender:

```
GSE0000I:  La operación se ejecutó satisfactoriamente.
```

Tabla 11. Partes del mensaje de DB2 Spatial Extender

Parte del mensaje	Descripción
GSE	Identificador del mensaje. Todos los mensajes de DB2 Spatial Extender comienzan con el prefijo de tres letras GSE.
0000	Número del mensaje. Es un número de cuatro dígitos comprendido entre 0000 y 9999.

Tabla 11. Partes del mensaje de DB2 Spatial Extender (continuación)

Parte del mensaje	Descripción
I	Tipo de mensaje. Es una letra individual que indica la gravedad del mensaje: **C** Mensajes de error críticos **N** Mensajes de error no críticos **W** Mensajes de aviso **I** Mensajes informativos
La operación se ejecutó satisfactoriamente.	Explicación del mensaje.

Es una breve descripción que aparece en el texto del mensaje. El usuario puede obtener más información sobre el mensaje, que incluye una explicación detallada y sugerencias para evitar o corregir el problema. Para visualizar esa información:

1. Abra un indicador de mandatos del sistema operativo.
2. Escriba el mandato de ayuda de DB2 junto con el identificador y número del mensaje para visualizar más información sobre el mensaje. Por ejemplo:

   ```
   DB2 "? GSEnnnn"
   ```

   donde *nnnn* es el número de mensaje.

Puede escribir el identificador del mensaje GSE y la letra que indica el tipo de mensaje en mayúsculas o minúsculas. Si escribe `DB2 "? GSE0000I"` obtendrá el mismo resultado que escribiendo `db2 "? gse0000i"`.

Puede omitir la letra que sigue al número de mensaje cuando escriba el mandato. Por ejemplo, si escribe `DB2 "? GSE0000"` obtendrá el mismo resultado que escribiendo `DB2 "? GSE0000I"`.

Suponga que el código de mensaje es `GSE4107N`. Si escribe `DB2 "? GSE4107N"` en el indicador de mandatos se mostrará la siguiente información:

```
GSE4107N  El valor de tamaño de retícula
<tamaño-retícula>" no es válido en el contexto utilizado.

Explicación: El tamaño de retícula especificado "<tamaño-retícula>" no
es válido.

 Se realizó una de las siguientes especificaciones no válidas
 cuando se creó el índice reticular con la sentencia CREATE INDEX:

  - Se ha especificado un número menor que 0 (cero) como tamaño de retícula
    para el primer, segundo o tercer nivel reticular.

  - Se ha especificado 0 (cero) como tamaño de retícula para el primer nivel
    de retícula.

  - El tamaño de retícula especificado para el segundo nivel reticular
    es menor que el tamaño de retícula del primer nivel, pero no es 0 (cero).

  - El tamaño de retícula especificado para el tercer nivel reticular
    es menor que el tamaño de retícula del segundo nivel, pero no es 0
    (cero).

  - El tamaño de retícula especificado para el tercer nivel reticular
    es mayor que 0 (cero), pero el tamaño de retícula especificado para
    el segundo nivel es 0 (cero).
```

```
Respuesta del usuario: especifique un valor válido para el tamaño de
retícula.

msgcode: -4107

sqlstate: 38SC7
```

Si la información es demasiado larga para visualizarse en una sola pantalla y su sistema operativo permite utilizar el mandato **more** y redireccionamientos de mandato, escriba:

```
db2 "? GSEnnnn" | more
```

La especificación **more** hace que la visualización haga una pausa después de cada pantalla de datos para que el usuario pueda leer la información.

Conceptos relacionados:

- "Parámetros de salida de los procedimientos almacenados de DB2 Spatial Extender" en la página 153
- "Mensajes de las funciones de DB2 Spatial Extender" en la página 155
- "Mensaje del CLP de DB2 Spatial Extender" en la página 157
- "Mensajes del Centro de control de DB2" en la página 159
- "Archivo de notificaciones de administración" en la página 161

Tareas relacionadas:

- "Rastreo de problemas de DB2 Spatial Extender con el mandato db2trc" en la página 160

Información relacionada:

- "Mensajes GSE" en el manual *Consulta de mensajes Volumen 1*

Parámetros de salida de los procedimientos almacenados de DB2 Spatial Extender

Los procedimientos almacenados de DB2® Spatial Extender se invocan *implícitamente* cuando habilita y utiliza Spatial Extender desde el Centro de control de DB2 o cuando utiliza el CLP de DB2 Spatial Extender (db2se). Puede invocar *explícitamente* a procedimientos almacenados desde un programa de aplicación o desde la línea de mandatos de DB2.

Este tema describe cómo diagnosticar problemas cuando se invocan explícitamente a procedimientos almacenados desde programas de aplicación o desde la línea de mandatos de DB2. Para diagnosticar procedimientos almacenados que se han invocado implícitamente, utilice los mensajes emitidos por el CLP de DB2 Spatial Extender o los mensajes emitidos por el Centro de control de DB2. Estos mensajes se tratan en temas separados.

Los procedimientos almacenados de DB2 Spatial Extender tienen dos parámetros de salida: el código del mensaje (msg_code) y el texto del mensaje (msg_text). Los valores de los parámetros indican si el procedimiento almacenado se ha ejecutado satisfactoriamente o no.

msg_code
El parámetro msg_code es un valor entero que puede ser positivo, negativo

o cero (0). Los valores positivos se utilizan para los avisos, los valores negativos se utilizan para los errores (críticos o no) y el cero (0) se utiliza para los mensajes informativos.

El valor absoluto de msg_code se incluye en msg_text y constituye el número del mensaje. Por ejemplo

- Si msg_code es 0, el número de mensaje es 0000.
- Si msg_code es –219, el número de mensaje es 0219. El valor negativo de msg_code indica que el mensaje indica un error, crítico o no.
- Si msg_code es +1036, el número de mensaje es 1036. El valor positivo de msg_code indica que el mensaje es un aviso.

Los valores de los códigos de mensaje emitidos por los procedimientos almacenados de Spatial Extender se clasifican en tres categorías, tal como muestra la tabla siguiente:

Tabla 12. Códigos de mensaje de procedimiento almacenado

Códigos	Categoría
0000 – 0999	Mensajes comunes
1000 – 1999	Mensajes administrativos
2000 – 2999	Mensajes de importación y exportación

msg_text

El parámetro msg_text comprende el identificador del mensaje, el número de mensaje, el tipo de mensaje y la explicación. El ejemplo siguiente muestra un valor de msg_text de un procedimiento almacenado:

```
GSE0219N   Ha fallado una sentencia
           EXECUTE IMMEDIATE. SQLERROR = "<error-sql>".
```

En el parámetro msg_text se incluye una breve explicación. El usuario puede obtener más información sobre el mensaje, que incluye una explicación detallada y sugerencias para evitar o corregir el problema.

Para conocer detalles sobre las partes del parámetro msg_text, e información sobre cómo recuperar más información sobre el mensaje, consulte el apartado ″Cómo interpretar los mensajes de DB2 Spatial Extender″.

Utilización de procedimientos almacenados en aplicaciones:

Cuando el usuario invoca a un procedimiento almacenado de DB2 Spatial Extender desde una aplicación, recibe el código y el texto del mensaje como parámetros de salida. El usuario puede:

- Programar la aplicación para devolver los valores de parámetros de salida al usuario de la aplicación.
- Emprender acciones de acuerdo con el tipo de valor de código de mensaje devuelto.

Utilización de procedimientos almacenados desde la línea de mandatos de DB2:

Cuando el usuario invoca a un procedimiento almacenado de DB2 Spatial Extender desde la línea de mandatos de DB2, recibe el código y el texto del mensaje como parámetros de salida. Estos parámetros indican si el procedimiento almacenado se ha ejecutado satisfactoriamente o no.

Suponga que se conecta a una base de datos y desea invocar al procedimiento almacenado ST_disable_db. El ejemplo siguiente utiliza un mandato CALL de DB2 para inhabilitar la base de datos para operaciones espaciales y muestra los resultados obtenidos. Se utiliza el parámetro force con un valor de 0, junto con dos signos de interrogación al final del mandato CALL para representar los parámetros de salida msg_code y msg_text. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

```
call db2gse.st_disable_db(0, ?, ?)

  Valor de los parámetros de salida
  ---------------------------------
  Nombre de parámetro : MSGCODE
  Valor de parámetro  : 0

  Nombre de parámetro : MSGTEXT
  Valor de parámetro  : GSE0000I  La operación se ejecutó satisfactoriamente.

  Estado de retorno = 0
```

Suponga que el valor devuelto de msg_text es GSE2110N. Utilice el mandato help de DB2 para visualizar más información sobre el mensaje. Por ejemplo:

```
"? GSE2110"
```

Se visualiza la información siguiente:

```
GSE2110N    El sistema de referencia espacial
            para la geometría de la fila "<número-fila>" no es válido.
            El identificador numérico del sistema de referencia espacial
            es "<id-srs>".

Explicación: En la fila número-fila, la geometría que se debe
exportar utiliza un sistema de referencia espacial no válido.
La geometría no se puede exportar.
Respuesta del usuario: Corrija la geometría indicada o
excluya la fila de la operación de exportación
modificando la sentencia SELECT según proceda.

msg_code: -2110

sqlstate: 38S9A
```

Conceptos relacionados:

- "Cómo interpretar los mensajes de DB2 Spatial Extender" en la página 151
- "Mensajes de las funciones de DB2 Spatial Extender" en la página 155
- "Mensaje del CLP de DB2 Spatial Extender" en la página 157
- "Mensajes del Centro de control de DB2" en la página 159

Información relacionada:

- "Mensajes GSE" en el manual *Consulta de mensajes Volumen 1*

Mensajes de las funciones de DB2 Spatial Extender

Los mensajes emitidos por las funciones de DB2® Spatial Extender suelen estar incluidos en un mensaje de SQL. El código de SQL (SQLCODE) devuelto en el mensaje indica se ha producido un error o aviso para la función. Por ejemplo:

- El SQLCODE –443 (número de mensaje SQL0443) indica que se ha producido un error en la función.

- El SQLCODE +462 (número de mensaje SQL0462) indica que existe una condición de aviso asociada a la función.

La tabla siguiente explica las partes significativas de este mensaje de ejemplo:

```
DB21034E  El mandato se ha procesado como una sentencia de SQL porque
no era un mandato válido para el procesador de línea de mandatos.
Durante el proceso de SQL se ha devuelto: SQL0443N  La rutina
"DB2GSE.GSEGEOMFROMWKT" (nombre específico "GSEGEOMWKT1") ha devuelto
un error
SQLSTATE con texto de diagnóstico "GSE3421N  El polígono no está cerrado.".
SQLSTATE=38SSL
```

Tabla 13. Partes significativas de los mensajes de las funciones de DB2 Spatial Extender

Parte del mensaje	Descripción
SQL0443N	El SQLCODE indica el tipo de problema.
GSE3421N	Es el número de mensaje y tipo de mensaje de DB2 Spatial Extender. Los números de mensaje de las funciones están comprendidos entre GSE3000 y GSE3999. Además, se pueden emitir mensajes comunes al trabajar con funciones de DB2 Spatial Extender. Los números de mensaje de los mensajes comunes están comprendidos entre GSE0001 y GSE0999.
Polígono no cerrado	Explicación del mensaje de DB2 Spatial Extender.
SQLSTATE=38SSL	Código SQLSTATE que identifica con más detalle el error. Se emite un código SQLSTATE para cada sentencia o fila. • Los códigos SQLSTATE para errores de funciones de Spatial Extender tienen la forma 38S*xx*, donde *x* es una letra o un número. • Los códigos SQLSTATE para avisos de funciones de Spatial Extender tienen la forma 01HS*x*, donde *x* es una letra o un número.

Ejemplo de mensaje de error QL0443:

Suponga que intenta insertar los valores de un polígono en la tabla POLYGON_TABLE, tal como se muestra a continuación:

```
INSERT INTO polygon_table ( geometry )
VALUES ( ST_Polygon ( 'polygon (( 0 0, 0 2, 2 2, 1 2)) ') )
```

Esto origina un mensaje de error debido a que el usuario no ha proporcionado el valor final para cerrar el polígono. El mensaje de error devuelto es:

```
DB21034E  El mandato se ha procesado como una sentencia de SQL porque
no era un mandato válido para el procesador de línea de mandatos. Durante
el proceso de SQL se ha devuelto: SQL0443N  La rutina
"DB2GSE.GSEGEOMFROMWKT" (nombre específico "GSEGEOMWKT1")
ha devuelto un error
SQLSTATE con texto de diagnóstico "GSE3421N  El polígono no está cerrado.".
SQLSTATE=38SSL
```

El número de mensaje de SQL `SQL0443N` indica que se ha producido un error y el mensaje incluye el texto del mensaje de Spatial Extender: `GSE3421N Polígono no cerrado`.

Si recibe este tipo de mensaje:

1. Localice el número de mensaje GSE dentro del mensaje de error de DB2 o SQL.

2. Utilice el mandato de ayuda de DB2 (DB2 ?) para ver la explicación del mensaje de Spatial Extender y la respuesta del usuario. Utilizando el ejemplo anterior, escriba el mandato siguiente en el indicador de línea de mandatos del sistema operativo:

   ```
   DB2 "? GSE3421"
   ```

El mensaje se repite, junto con una explicación detallada y la respuesta del usuario recomendada.

Conceptos relacionados:

- "Cómo interpretar los mensajes de DB2 Spatial Extender" en la página 151
- "Parámetros de salida de los procedimientos almacenados de DB2 Spatial Extender" en la página 153
- "Mensaje del CLP de DB2 Spatial Extender" en la página 157
- "Mensajes del Centro de control de DB2" en la página 159

Información relacionada:

- "Mensajes GSE" en el manual *Consulta de mensajes Volumen 1*

Mensaje del CLP de DB2 Spatial Extender

El CLP de DB2® Spatial Extender (db2se) emite mensajes para:

- Procedimientos almacenados, si se invocan implícitamente.
- Información sobre formas, si ha invocado al programa de submandato **shape_info** desde el CLP de DB2 Spatial Extender. Estos mensajes son informativos.
- Operaciones de migración.
- Operaciones de importación y exportación de formas con el cliente.

Ejemplos de mensajes de procedimiento almacenado emitidos por el CLP de DB2 Spatial Extender:

La mayoría de los mensajes emitidos desde el CLP de DB2 Spatial Extender corresponden a procedimientos almacenados de DB2 Spatial Extender. Cuando invoca a un procedimiento almacenado desde el CLP de DB2 Spatial Extender, recibe un texto de mensaje que indica el éxito o fracaso del procedimiento almacenado.

El texto del mensaje comprende el identificador del mensaje, el número de mensaje, el tipo de mensaje y la explicación. Por ejemplo, si habilita una base de datos utilizando el mandato `db2se enable_db testdb`, el texto del mensaje devuelto por el CLP de Spatial Extender es:

```
Se está habilitando la base de datos. Espere, por favor ...

GSE1036W   La operación se ha realizado satisfactoriamente.  Sin
           embargo, los valores de determinados parámetros del gestor
           de base de datos y de configuración de base de datos
           deberían aumentarse.
```

Del mismo modo, si habilita una base de datos utilizando el mandato `db2se disable_db testdb`, el texto del mensaje devuelto por el CLP de Spatial Extender es:

```
GSE0000I  La operación se ejecutó satisfactoriamente.
```

Es una breve descripción que aparece en el texto del mensaje. El usuario puede obtener más información sobre el mensaje, que incluye una explicación detallada y sugerencias para evitar o corregir el problema. Los pasos para recuperar esta información, e información detallada sobre cómo interpretar las partes del texto del mensaje, se tratan en un tema separado.

Si está invocando a procedimientos almacenados desde un programa de aplicación o desde la línea de mandatos de DB2, en un tema separado se trata el diagnóstico de los parámetros de salida.

Ejemplo de mensajes de información sobre formas emitidos por el CLP de Spatial Extender:

Suponga que desea visualizar información sobre un archivo de formas llamado `office`. Desde el CLP de Spatial Extender (db2se) emitiría este mandato:

```
db2se shape_info -nombreArchivo /tmp/offices
```

Esto es un ejemplo de la información que se visualiza:

```
Información del archivo de formas
---------------------------------
Código de archivo                          = 9994
Longitud del archivo (palabras de 16 bits) = 484
Versión del archivo de formas              = 1000
Tipo de forma                              = 1 (ST_POINT)
Número de registros                        = 31

Coordenada X mínima  = -87.053834
Coordenada X máxima  = -83.408752
Coordenada Y mínima  = 36.939628
Coordenada Y máxima  = 39.016477
Las formas no tienen coordenadas Z.
Las formas no tienen coordenadas M.

Aparece el archivo de índices de formas (extensión .shx).

Información del archivo de atributos
------------------------------------
Código del archivo dBase          = 3
Fecha de la última actualización  = 1901-08-15
Número de registros               = 31
Número de bytes en la cabecera    = 129
Número de bytes en cada registro  = 39
Número de columnas                = 3

Número de columna  Nombre de columna      Tipo de dato       Longitud  Decimal

                1  NAME                   C (Character)      16        0
                2  EMPLOYEES              N (Numeric™)    11        0
                3  ID                     N (Numeric)        11        0

Definición del sistema de coordenadas: "GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433295]]"
```

Ejemplo de mensajes de migración emitidos por el CLP de Spatial Extender:

Cuando el usuario invoca a mandatos que realizan operaciones de migración, se emiten mensajes que indican el éxito o el fracaso de esa operación.

Suponga que ejecuta el mandato `db2se migrate mi_bd -archivoMensajes /tmp/migrate.msg` para migrar la base de datos `mi_bd`. El texto del mensaje devuelto por el CLP de Spatial Extender es:

```
Se está migrando la base de datos. Espere, por favor ...
GSE0000I  La operación se ejecutó satisfactoriamente.
```

Conceptos relacionados:

- "Cómo interpretar los mensajes de DB2 Spatial Extender" en la página 151
- "Parámetros de salida de los procedimientos almacenados de DB2 Spatial Extender" en la página 153
- "Mensajes de las funciones de DB2 Spatial Extender" en la página 155
- "Mensajes del Centro de control de DB2" en la página 159

Información relacionada:

- "Mensajes GSE" en el manual *Consulta de mensajes Volumen 1*

Mensajes del Centro de control de DB2

Cuando el usuario utiliza DB2® Spatial Extender mediante el Centro de control de DB2, se muestran mensajes en la ventana de mensajes de DB2. La mayoría de los mensajes que recibe el usuario son mensajes de DB2 Spatial Extender. Ocasionalmente, el usuario recibe un mensaje de SQL. Se emiten mensajes de SQL cuando se produce un error referente al uso de licencias, bloqueos, o cuando un servicio DAS no está disponible. Las secciones siguientes proporcionan ejemplos de cómo aparecen los mensajes de DB2 Spatial Extender y de SQL en el Centro de control de DB2.

Mensajes de DB2 Spatial Extender:

Cuando recibe un mensaje de DB2 Spatial Extender a través del Centro de control, el texto completo del mensaje aparece en el área de texto de la ventana de mensajes de DB2, por ejemplo:

```
GSE0219N   Ha fallado una sentencia
           EXECUTE IMMEDIATE. SQLERROR = "<error-sql>".
```

Mensajes de SQL:

Cuando recibe un mensaje de SQL a través del Centro de control que pertenece a DB2 Spatial Extender:

- El identificador, número y tipo del mensaje aparecen en el lado izquierdo de la ventana de mensajes de DB2, por ejemplo: `SQL0612N`.
- El texto del mensaje aparece en el área de texto de la ventana de mensajes de DB2.

El texto del mensaje que aparece en la ventana de mensajes de DB2 puede incluir el texto de mensaje de SQL y el SQLSTATE, o bien puede contener el texto del mensaje, la explicación detallada y la respuesta del usuario.

Esto es un ejemplo de un mensaje de SQL que contiene el texto del mensaje de SQL y el SQLSTATE:

```
[IBM][CLI Driver][DB2/NT] SQL0612N "<nombre>" es un nombre
duplicado. SQLSTATE=42711
```

Esto es un ejemplo de un mensaje de SQL que contiene el texto del mensaje, la explicación detallada y la respuesta del usuario:

```
SQL8008N
El producto "DB2 Spatial Extender" no tiene instalada una clave
de licencia válida y el período de evaluación ha terminado.

Explicación:

No se ha podido encontrar ninguna clave de licencia válida y el
período de evaluación ha caducado.

Respuesta del usuario:

Instale una clave de licencia para la versión del producto totalmente
autorizada.  Puede obtener una clave de licencia para el producto poniéndose
en contacto con el representante de IBM® o el concesionario autorizado.
```

Conceptos relacionados:

- "Cómo interpretar los mensajes de DB2 Spatial Extender" en la página 151
- "Parámetros de salida de los procedimientos almacenados de DB2 Spatial Extender" en la página 153
- "Mensajes de las funciones de DB2 Spatial Extender" en la página 155
- "Mensaje del CLP de DB2 Spatial Extender" en la página 157

Información relacionada:

- "Mensajes GSE" en el manual *Consulta de mensajes Volumen 1*

Rastreo de problemas de DB2 Spatial Extender con el mandato db2trc

Si se produce un problema repetitivo y reproducible de DB2 Spatial Extender, puede utilizar el recurso de rastreo de DB2 para recoger información sobre el problema. El recurso de rastreo de DB2 se activa con el mandato del sistema **db2trc**. El recurso de rastreo de DB2 puede:

- Rastrear sucesos
- Volcar los datos de rastreo en un archivo
- Formatear los datos de rastreo para convertirlos en un formato legible.

Restricciones:

Active este recurso sólo cuando se lo indique el servicio técnico de DB2.

En los sistemas operativos UNIX, debe tener autorización SYSADM, SYSCTRL o SYSMAINT para rastrear una instancia de DB2.

En los sistemas operativos Windows no es necesaria ninguna autorización especial.

Procedimiento:

Para rastrear sucesos de DB2 Spatial Extender en la memoria, siga estos pasos básicos:

1. Cierre todas las demás aplicaciones.
2. Active el recurso de rastreo. El servicio técnico de DB2 le proporcionará los parámetros específicos para este paso. El mandato básico es:

   ```
   db2trc on
   ```

Restricción: El mandato **db2trc** se debe entrar en un indicador del sistema operativo o especificar en un script de shell. No se puede utilizar en la interfaz de mandatos de DB2 Spatial Extender (db2se) ni en el CLP de DB2.

Puede volcar datos de rastreo en la memoria o en un archivo. El método preferido es volcar datos de rastreo en la memoria. Si el programa que se está reproduciendo suspende la actividad de la estación de trabajo y le impide volcar el rastreo, realice el rastreo en un archivo.

3. Reproduzca el problema.
4. Vuelque el rastreo en un archivo. Por ejemplo:

   ```
   db2trc dump january23trace.dmp
   ```

   Este mandato crea un archivo (*january23trace.dmp*) en el directorio actual con el nombre especificado por el usuario y vuelca la información de rastreo en ese archivo.

   Puede especificar un directorio diferente indicando la vía de acceso del archivo. Por ejemplo, para colocar el archivo de rastreo en el directorio /tmp/spatial/errors, la sintaxis es:

   ```
   db2trc dump /tmp/spatial/errors/january23trace.dmp
   ```

   Vuelque el rastreo inmediatamente después de producirse el problema.
5. Desactive el rastreo. Por ejemplo:

   ```
   db2trc off
   ```
6. Formatee los datos como archivo ASCII. Puede clasificar los datos de dos manera:
   - Utilice la opción **flw** para clasificar los datos según el proceso o hebra. Por ejemplo:

     ```
     db2trc flw january23trace.dmp january23trace.flw
     ```
   - Utilice la opción **fmt** para listar los sucesos cronológicamente. Por ejemplo:

     ```
     db2trc fmt january23trace.dmp january23trace.fmt
     ```

Conceptos relacionados:

- "DB2 trace (db2trc)" en el manual *Troubleshooting Guide*
- "Cómo interpretar los mensajes de DB2 Spatial Extender" en la página 151
- "Archivo de notificaciones de administración" en la página 161

Información relacionada:

- "Mensajes GSE" en el manual *Consulta de mensajes Volumen 1*

Archivo de notificaciones de administración

La información de diagnóstico acerca de errores se registra en el archivo de notificaciones de administración. Este información se utiliza para la determinación de problemas y está pensada para el soporte técnico de DB2®.

El archivo de notificaciones de administración contiene información de texto registrada por DB2 y por DB2 Spatial Extender. Está ubicado en el directorio especificado por el parámetro de configuración del gestor de la base de datos DIAGPATH. En los sistemas Windows® NT, Windows 2000 y Windows XP, el archivo de notificaciones de administración de DB2 se encuentra en el archivo de anotaciones cronológicas de sucesos y se puede examinar mediante el Visor de Sucesos de Windows.

La información que DB2 registra en el archivo de anotaciones cronológicas de administración está determinada por los valores DIAGLEVEL y NOTIFYLEVEL.

Utilice un editor de texto para visualizar el archivo en la máquina donde sospecha que se ha producido un problema. Los sucesos más recientes aparecen registrados al final del archivo. Generalmente, cada entrada del archivo tiene estas partes:

- Indicación de fecha y hora.
- La ubicación que notifica el error. Los identificadores de aplicación permiten al usuario emparejar entradas pertenecientes a una aplicación en los archivos de anotaciones cronológicas de servidores y clientes.
- Un mensaje de diagnóstico (que generalmente comienza por ″DIA″ o ″ADM″) que describe el error.
- Datos auxiliares disponibles, tales como estructuras de datos SQLCA y punteros que indican la posición de archivos de vuelco o de captura adicionales.

Si el comportamiento de la base de datos es normal, este tipo de información no es importante y se puede pasar por alto.

El archivo de notificaciones de administración crece constantemente. Cuando su tamaño sea demasiado grande, haga una copia del archivo y luego borre el archivo. El sistema generará automáticamente un nuevo archivo la próxima vez que sea necesario.

Conceptos relacionados:

- “Interpreting the administration logs” en el manual *Troubleshooting Guide*
- “Cómo interpretar los mensajes de DB2 Spatial Extender” en la página 151

Tareas relacionadas:

- “Rastreo de problemas de DB2 Spatial Extender con el mandato db2trc” en la página 160

Información relacionada:

- “Mensajes GSE” en el manual *Consulta de mensajes Volumen 1*

Parte 4. Utilización de DB2 Geodetic Extender

Capítulo 16. DB2 Geodetic Extender

Este capítulo proporciona información preliminar sobre DB2 Geodetic Extender; describe su función, cuándo se utiliza y explica conceptos geodésicos.

DB2 Geodetic Extender

DB2® Geodetic Extender permite tratar la tierra como si fuera un globo. Utilizando los mismos tipos de datos y funciones espaciales que para el resto de las operaciones de Spatial Extender, puede utilizar Geodetic Extender para ejecutar consultas transparentes de datos transpolares y de datos que cruzen el meridiano 180. Puede realizar el mantenimiento de datos que hagan referencia a una ubicación precisa sobre la superficie de la tierra.

Geodetic Extender debe su nombre a la disciplina denominada *geodesia*. La geodesia es el estudio del tamaño y de la forma de la Tierra (o de cualquier cuerpo modelado por un elipsoide, como el sol o una esfera celeste). El diseño de Geodetic Extender permite manejar objetos definidos en la superficie de la tierra con un alto grado de precisión.

Para lograr esta precisión, Geodetic Extender utiliza un sistema de coordenadas de latitud y longitud en un modelo de la tierra elipsoidal, o *datum geodésico*, en lugar de un sistema de coordenadas *x*- e *y*- plano. Un modelo elipsoidal evita las distorsiones, inexactitudes e imprecisiones que pueden producir las proyecciones planas. Para obtener más información, consulte los apartados "Latitud y longitud geodésica" en la página 167, "Sistema de coordenadas geográficas" en la página 60 y "Sistemas de coordenadas proyectadas" en la página 65.

Para acceder a operaciones geodésicas en vez de espaciales, deberá definir un sistema de referencia espacial geodésico para sus datos. Estos sistemas tienen ID de sistemas de referencia espacial (SRID) comprendidos entre 2000000000 y 2000001000. Geodetic Extender proporciona 318 sistemas de referencia espacial geodésicos predefinidos.

Para utilizar DB2 Geodetic Extender debe instalar DB2 Spatial Extender. El pedido de DB2 Geodetic Extender se realiza independientemente del de DB2 Spatial Extender, y deberá adquirir una licencia separada para Geodetic Extender.

Conceptos relacionados:

- "Cuándo utilizar DB2 Geodetic Extender y DB2 Spatial Extender" en la página 166
- "Puntos de referencia geodésicos" en la página 166

Tareas relacionadas:

- "Configuración y habilitación de DB2 Geodetic Extender" en la página 173

Información relacionada:

- "Sistemas de referencia soportados por DB2 Geodetic Extender" en la página 220

Cuándo utilizar DB2 Geodetic Extender y DB2 Spatial Extender

Tanto DB2® Spatial Extender como DB2 Geodetic Extender gestionan datos de sistemas de información geográfica (GIS) en una base de datos de DB2. Cada extensor utiliza distintas tecnologías básicas para resolver diferentes problemas y se complementan mutuamente:

- Geodetic Extender considera que la tierra es un globo. Utiliza un sistema de coordenadas de latitud y longitud en un modelo elipsoidal de la tierra. Las operaciones geométricas son precisas, independientemente de la ubicación. Se basa en la biblioteca Hypparchus, cuya licencia es propiedad de Geodyssey Limited. Consulte http://www.geodyssey.com para obtener más información geodésica.

  Geodetic Extender se usa principalmente con aplicaciones y conjuntos de datos globales que cubren áreas grandes de la tierra, en los que una simple proyección de mapa no puede proporcionar la precisión que requiere la aplicación.
- Spatial Extender considera que la tierra es un mapa plano. Utiliza la geometría planimétrica (de nivel plano), es decir, que se aproxima a la superficie redonda de la tierra proyectándola en un nivel plano. Esta proyección produce distorsiones, que pueden variar en función de la extensión de los datos, aunque las distorsiones aumentan generalmente hacia los bordes de la región proyectada. Todas las proyecciones de nivel plano sufren algún tipo de distorsión. Spatial Extender se basa en la biblioteca de formas ESRI, cuya licencia es propiedad de ESRI. Consulte http://www.esri.com para obtener más información espacial.

  Spatial Extender se utiliza principalmente para conjuntos de datos regionales y locales que se representan correctamente en coordenadas proyectadas y para aplicaciones en las que la precisión de la ubicación no reviste importancia. Por ejemplo, cuando una compañía de seguros médicos desea conocer las ubicaciones de los hospitales y las clínicas de una región o provincia determinada.

Conceptos relacionados:

- "DB2 Geodetic Extender" en la página 165
- "Regiones geodésicas" en la página 170
- "Latitud y longitud geodésica" en la página 167
- "Distancias geodésicas" en la página 168
- "Esferoides geodésicos" en la página 229

Tareas relacionadas:

- "Configuración y habilitación de DB2 Geodetic Extender" en la página 173

Puntos de referencia geodésicos

Un datum geodésico es un sistema de referencia que describe la superficie de la tierra. Durante siglos se han desarrollado muchos sistemas de referencia de este tipo a medida que la ciencia ha ido desarrollando nuevas herramientas para medir la tierra. Se han utilizado mediciones terrestres y por satélite para crear datums, que a su vez se han destinado a crear proyecciones de nivel plano.

Los datums geodésicos se basan en una aproximación de la forma general de la tierra mediante un elipsoide de rotación (también denominada *esferoide*). Un esferoide es la forma tridimensional que describe una elipse cuando gira alrededor

de uno de sus ejes. Para obtener más información sobre los esferoides, consulte el apartado "Sistema de coordenadas geográficas" en la página 60.

Todos los objetos espaciales que defina deben hacer alusión a un datum específico. Los datums se especifican mediante identificadores de sistemas de referencia espacial (SRID). Puede elegir cualquier datum al que DB2® Geodetic Extender dé soporte. Estos sistemas tienen SRID comprendidos entre el 2000000000 y el 2000001000.

- El apartado "Sistemas de referencia soportados por DB2 Geodetic Extender" en la página 220 enumera los 318 sistemas de referencia espacial geodésicos predefinidos que proporciona Geodetic Extender.
- También puede definir un nuevo datum creando un sistema de referencia espacial con un ID comprendido entre el 2000000318 y el 2000001000. Para obtener más información, consulte el apartado "Creación de un sistema de referencia espacial" en la página 76.

Restricciones: Las funciones que toman más de un objeto espacial como argumento no pueden manejar combinaciones de puntos de referencia. Geodetic Extender no realiza conversiones de puntos de referencia.

Conceptos relacionados:

- "Cuándo utilizar DB2 Geodetic Extender y DB2 Spatial Extender" en la página 166
- "Regiones geodésicas" en la página 170
- "Latitud y longitud geodésica" en la página 167
- "Distancias geodésicas" en la página 168
- "Esferoides geodésicos" en la página 229

Tareas relacionadas:

- "Cómo decidirse entre la utilización de un sistema de referencia espacial por omisión o la creación de un sistema nuevo" en la página 70
- "Creación de un sistema de referencia espacial" en la página 76

Información relacionada:

- "Sistemas de referencia soportados por DB2 Geodetic Extender" en la página 220

Latitud y longitud geodésica

El sistema de referencia de coordenadas de DB2 Geodetic Extender utiliza la *latitud* y la *longitud* geodésica para describir ubicaciones con relación a la tierra. La latitud y la longitud geodésica se basan siempre en un datum específico.

Latitud geodésica
La latitud geodésica de un punto es el ángulo que forman el plano ecuatorial y la línea perpendicular que cruza la línea normal en el punto de la superficie de la tierra.

Longitud geodésica
La longitud geodésica es el ángulo en el plano ecuatorial que forma la línea *a* que conecta el centro de la tierra con el meridiano de origen con la línea *b* que conecta el centro con el meridiano en el que reside el punto. Un *meridiano* es una trayectoria directa sobre la superficie del datum que constituye la distancia más corta entre los polos.

El elipsoide de la Figura 17 muestra los ángulos que representan la latitud y la longitud geodésica. El ángulo de la latitud geodésica no empieza en el mismo centro debido a la forma elipsoidal de la tierra.

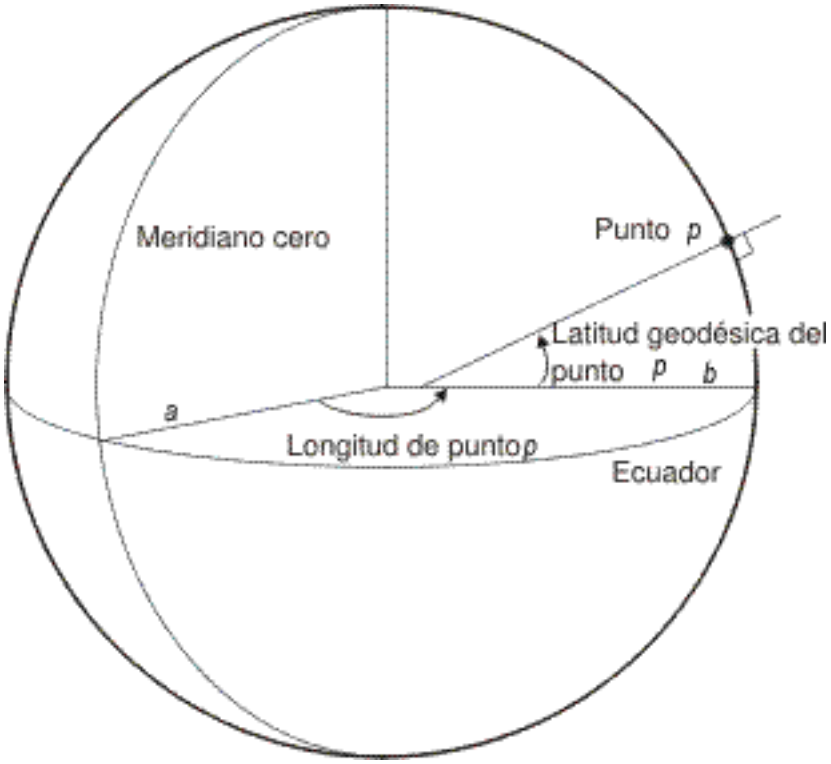


Figura 17. Ángulos de latitud y longitud geodésica

Las coordenadas de latitud y longitud se expresan en grados con una fracción decimal. Hay 360 grados de longitud que empiezan en el meridiano de origen (longitud 0°) y siguen en dirección oeste en sentido positivo hasta los 180° y hacia el oeste en valores negativos hasta los –180°. Los grados de latitud empiezan en el ecuador (latitud 0°) y siguen hacia el polo norte (latitud 90°) y hacia el polo sur (latitud –90°).

Conceptos relacionados:

- "DB2 Geodetic Extender" en la página 165
- "Cuándo utilizar DB2 Geodetic Extender y DB2 Spatial Extender" en la página 166
- "Regiones geodésicas" en la página 170
- "Puntos de referencia geodésicos" en la página 166
- "Distancias geodésicas" en la página 168
- "Esferoides geodésicos" en la página 229

Distancias geodésicas

DB2® Geodetic Extender calcula la distancia entre dos puntos a lo largo de un *geodésico*. Un geodésico es la trayectoria más corta entre dos puntos de la forma elipsoidal de la tierra; es posible que esta trayectoria más corta no siga una latitud constante aunque los dos puntos finales estén en la misma latitud.

Puesto que los segmentos lineales se computan como geodésicos, es posible que un polígono de cuatro puntos con los puntos ampliamente separados no circunde la región de interés, tal como muestra la Figura 18. Este polígono cubre una región con líneas de longitud que están separadas unos 120 grados; los dos puntos superiores tienen los mismos valores de latitud y los dos puntos inferiores también. La geodésica entre las dos líneas de longitud sigue la curva de la forma elipsoidal de la tierra. La latitud se incrementa a lo largo de la geodésica hasta 20 grados más por el medio que en los extremos de la misma.

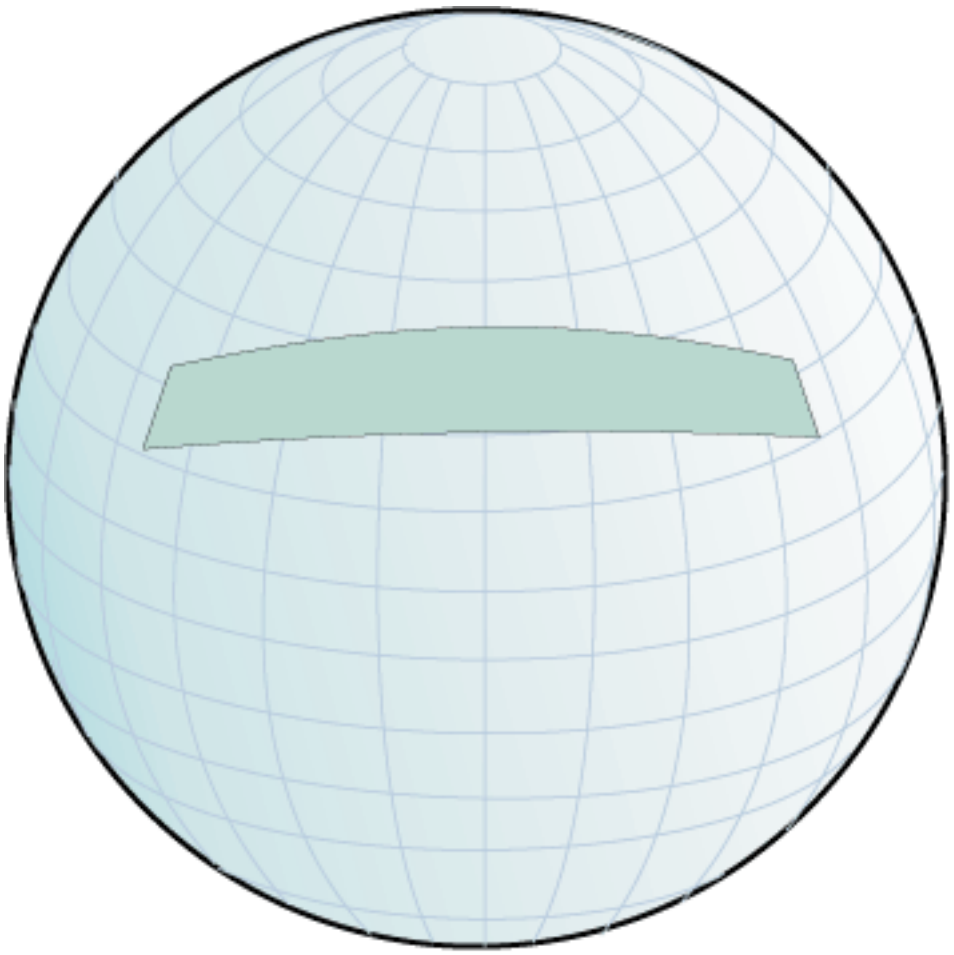


Figura 18. Región delimitada por un polígono con puntos ampliamente separados

Si desea representar una trayectoria que no sea geodésica, como por ejemplo, un segmento lineal que siga una latitud constante, deberá insertar puntos intermedios adicionales.

Conceptos relacionados:

- "Sistema de coordenadas geográficas" en la página 60

Información relacionada:

- "Diferencias de utilización entre las representaciones terrestres planas y redondas" en la página 205
- "ST_Distance" en la página 382

Regiones geodésicas

Una región geodésica (polígono) es un área de la superficie de la tierra que tiene alguna característica específica para una aplicación. Ejemplos de regiones podrían ser un área de influencia comercial o el área avistada por un satélite durante un periodo de tiempo determinado.

Geodetic Extender define una región mediante una secuencia ordenada de puntos que forman un anillo cerrado. El orden en que se especifican los puntos de un polígono es significativo. Siguiendo un polígono de vértice a vértice en el orden definido, el área situada a la izquierda se encuentra dentro del polígono.

Puede utilizar un tipo de dato ST_Polygon para definir una región delimitada por uno o varios anillos, tal como se muestra en la Figura 19 en la página 170. Defina el polígono mediante las coordenadas de latitud y longitud de los puntos (vértices) que forman sus anillos.

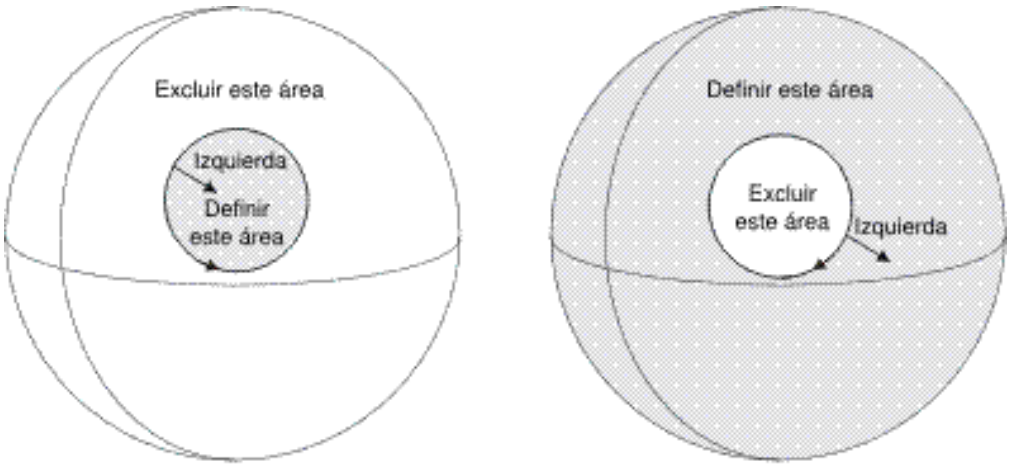


Figura 19. Áreas de definición y de exclusión

Un anillo divide la superficie de la tierra en dos regiones: una región dentro del polígono y otra fuera del polígono. La parte izquierda de la Figura 19 muestra un anillo con vértices especificados en orden contrario al de las agujas del reloj, de modo que todos los puntos de la izquierda están dentro del anillo. La parte derecha de la figura muestra un anillo cuyos vértices están especificados en orden contrario al de las agujas del reloj, de modo que todos los puntos de la izquierda están fuera del anillo.

Para definir una región como un polígono, debe especificar el orden de los vértices de cada anillo de modo que el interior del polígono esté a su izquierda cuando atraviese el anillo. Para definir una región excluida, debe especificar los vértices del anillo en orden inverso, tal como muestra la Figura 20 en la página 171. El interior del polígono siempre se encuentra a la izquierda. La Figura 20 en la página 171 muestra dos anillos, uno dentro del otro. El anillo más grande define el perímetro exterior del polígono y se ha dibujado en sentido contrario al de las agujas del reloj. El anillo más pequeño define el perímetro interno y se ha dibujado en el sentido de las agujas del reloj.

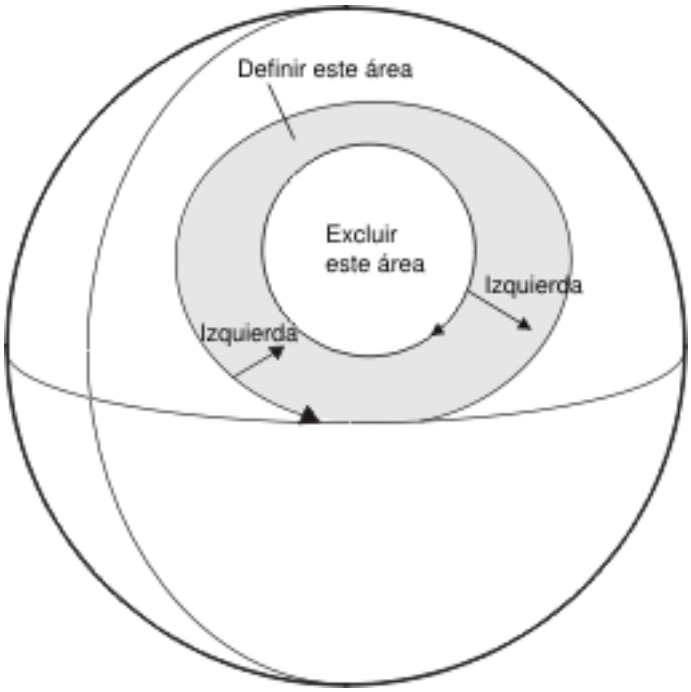


Figura 20. Definición de un área con varios anillos

Si crea un polígono más grande que un hemisferio, recibirá el siguiente mensaje de aviso. Es posible que desee conservar este polígono grande, pero el aviso aparece por si ha especificado sin advertirlo un orden equivocado en los vértices y en vez de un polígono pequeño ha generado un polígono grande.

```
GSE3733W "El polígono cubre más de la mitad de la tierra.
Compruebe la orientación de los puntos del vértice que sigue el sentido
de las agujas del reloj."
```

Conceptos relacionados:

- "Sistemas de coordenadas proyectadas" en la página 65
- "Puntos de referencia geodésicos" en la página 166
- "Sistemas de referencia espacial" en la página 68

Tareas relacionadas:

- "Cómo decidirse entre la utilización de un sistema de referencia espacial por omisión o la creación de un sistema nuevo" en la página 70
- "Creación de un sistema de referencia espacial" en la página 76

Capítulo 17. Configuración de DB2 Geodetic Extender

Este capítulo proporciona información sobre cómo configurar DB2 Geodetic Extender, migrar desde Informix Geodetic DataBlade y llenar columnas espaciales con datos geodésicos.

Configuración y habilitación de DB2 Geodetic Extender

DB2 Geodetic Extender considera que la tierra es un globo; si embargo, Spatial Extender considera que la superficie de la tierra es un mapa plano. Si instala Geodetic Extender, podrá analizar los datos espaciales con mayor precisión que un mapa plano.

Un sistema DB2 Geodetic Extender consta de DB2 Universal Database, DB2 Spatial Extender, DB2 Geodetic Extender y, para la mayoría de las aplicaciones, de un geonavegador.

Recomendación: Para consultar información adicional o modificada sobre la habilitación de DB2 Geodetic Extender, consulte el manual *Notas del release de DB2*.

Requisitos previos:

Para habilitar DB2 Geodetic Extender, deberá:

- Instalar y configurar DB2 Universal Database™ Enterprise Server Edition Versión 8.2.

  Debe instalar DB2 UDB en el sistema *antes* de instalar DB2 Spatial Extender y DB2 Geodetic Extender. Si piensa utilizar el Centro de control de DB2, cree y configure el Servidor de administración de DB2 (DAS). Para obtener más información sobre la creación y configuración del DAS, consulte el manual *IBM® DB2 Universal Database Administration Guide: Implementation*
- Instale y configure DB2 Spatial Extender.

  DB2 Geodetic Extender se integra en el mismo código de bibliotecas que DB2 Spatial Extender. Por lo tanto, el CD de instalación de Spatial Extender incluye Geodetic Extender. Los requisitos de espacio en disco para Spatial Extender incluyen Geodetic Extender. Sin embargo, no podrá utilizar Geodetic Extender hasta que adquiera y habilite una licencia de Geodetic Extender. Para obtener más información, consulte los apartados “Requisitos del sistema para la instalación de Spatial Extender” en la página 26 y “Instalación y configuración de Spatial Extender” en la página 25.
- Si tiene una base de datos de DB2 Spatial Extender Versión 8.1, deberá migrarla a la Versión 8.2 para poder utilizar DB2 Geodetic Extender.

  Geodetic Extender redefine varias funciones espaciales y define sistemas de referencia espacial geodésicos adicionales para manejar datos geodésicos. El programa de utilidad de migración **migrate_v82** permite manejar datos geodésicos a una base de datos habilitada para datos espaciales existente. Para obtener más información, consulte el apartado “Migración de una base de datos habilitada para operaciones espaciales” en la página 45.
- Adquiera una licencia de DB2 Geodetic Extender.

Cuando adquiera una licencia de DB2 Geodetic Extender podrá habilitar la clave de licencia de Geodetic. Póngase en contacto con su representante de ventas si desea adquirir DB2 Geodetic Extender.

Restricciones:

DB2 Geodetic Extender tiene licencia únicamente para DB2 Universal Database™ Enterprise Server Edition Versión 8.2.

Procedimiento:

Habilite la licencia de DB2 Geodetic Extender de una de las maneras siguientes:

- Utilice el Centro de licencias del Centro de control de DB2. Consulte la ayuda en línea del Centro de licencias de DB2 para obtener más información sobre cómo habilitar la licencia de Geodetic.
- Ejecute el mandato **db2licm**.

Tras habilitar la licencia de DB2 Geodetic Extender, deberá "Llenado de columnas espaciales con datos geodésicos" en la página 181.

Conceptos relacionados:

- "Requisitos del sistema para la instalación de Spatial Extender" en la página 26

Tareas relacionadas:

- "Instalación y configuración de Spatial Extender" en la página 25
- "Migración de una base de datos habilitada para operaciones espaciales" en la página 45
- "Llenado de columnas espaciales con datos geodésicos" en la página 181

Información relacionada:

- "CD para datos y mapas de DB2 Spatial Extender" en la página 43

Migración de Informix Geodetic DataBlade a DB2 Geodetic Extender

Si utiliza IBM Informix Geodetic DataBlade para almacenar y manipular objetos geoespaciales en una base de datos, puede migrar los datos y las aplicaciones a IBM DB2 Geodetic Extender con algunas restricciones.

Requisitos previos:

Debe transferir las aplicaciones de Geodetic DataBlade para utilizar los tipos de datos y las funciones de DB2 Geodetic Extender.

Restricciones:

Si actualmente utiliza Informix Geodetic DataBlade, probablemente podrá migrar a DB2 Geodetic Extender si satisface los siguientes criterios:

- Utiliza sólo tipos de datos GeoPoint, GeoLineseg, GeoString, GeoRing y GeoPolygon.
- Utiliza sólo funciones de Geodetic DataBlade que tienen su equivalente exacto o aproximado en DB2 Geodetic Extender, tal como describen las tablas que se muestran más adelante.

- Indexa sólo el componente espacial de GeoObjects; en otras palabras, no indexa rangos de tiempo ni rangos de altitud.

Procedimiento:

Para migrar de IBM Informix Geodetic DataBlade a IBM DB2 Geodetic Extender:

1. Rescriba las sentencias de SQL para utilizar tipos de datos y funciones de DB2 Geodetic Extender. Consulte en las tablas siguientes los tipos de datos y las funciones equivalentes:
   - Tabla 14
   - Tabla 15 en la página 176
   - Tabla 16 en la página 176
   - Tabla 17 en la página 177
   - Tabla 18 en la página 178
   - Tabla 19 en la página 178
   - Tabla 20 en la página 178
   - Tabla 21 en la página 179
   - Tabla 22 en la página 179
   - Tabla 23 en la página 179
2. Cargue o importe sus datos en DB2 Geodetic Extender.
3. Rescriba las aplicaciones que utilizan Informix ODBC, ESQL/C y JDBC. La Tabla 24 en la página 181 muestra la conectividad de cliente equivalente en Geodetic DataBlade y Geodetic Extender.

Tabla 14. Tipos de datos equivalentes en Informix Geodetic DataBlade y Geodetic Extender

Tipo de datos de Informix Geodetic DataBlade	Tipo de datos equivalente en DB2 Geodetic Extender	Comentarios para tipos de datos casi equivalentes
GeoBox		En primer lugar, conviértalo en un GeoPolygon en Geodetic DataBlade y, a continuación, utilice ST_Polygon en Geodetic Extender
GeoCircle		En primer lugar, conviértalo en un GeoPolygon y, a continuación, mígrelo a ST_Polygon
GeoEllipse		En primer lugar, conviértalo en un GeoPolygon y, a continuación, mígrelo a ST_Polygon
GeoLineseg	ST_LineString	
GeoObject	ST_Geometry	ST_Geometry y sus subtipos no tienen soporte para los tipos de datos GeoAltRange y GeoTimeRange
GeoPoint	ST_Point	
GeoPolygon	ST_MultiPolygon, ST_Polygon	ST_MultiPolygon requiere un punto de cierre explícito para cada anillo. Si un GeoPolygon tiene un anillo exterior, puede correlacionarse con un ST_Polygon.
GeoRing	ST_LineString	
GeoString	ST_LineString	

Los siguientes tipos de datos de Geodetic DataBlade no tienen tipos de datos equivalentes en Geodetic Extender:

- GeoAltitude

- GeoAltRange
- GeoAngle
- GeoAzimuth
- GeoBox
- GeoCircle
- GeoCoords
- GeoDistance
- GeoEllipse
- GeoLatitude
- GeoLongitude
- GeoTimeRange
- GeoVoronoi

Tabla 15. Funciones de predicado equivalentes en Informix Geodetic DataBlade y Geodetic Extender

Función de Informix Geodetic DataBlade	Función equivalente en DB2 Geodetic Extender
Contains	ST_Contains
Inside	ST_Within
Intersect	ST_Intersects
Outside	ST_Disjoint
Within	ST_Distance

Las siguientes funciones de predicado de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- Beyond
- Equal
- Nearest

Tabla 16. Funciones de producción equivalentes en Informix Geodetic DataBlade y Geodetic Extender

Función de Informix Geodetic DataBlade	Función equivalente en DB2 Geodetic Extender	Comentarios para tipos de datos casi equivalentes
Difference	ST_Difference	ST_Difference tiene soporte para puntos además de polígonos
Generalize	ST_Generalize	
Intersection	ST_Intersection	ST_Intersection(line,line) podría producir como resultado un multipunto. ST_Intersection (line,poly) podría producir como resultado una multilínea. Devuelve el valor Empty para los objetos inconexos.
SymDifference	ST_SymDifference	ST_SymDifference tiene soporte para puntos además de polígonos

Tabla 16. Funciones de producción equivalentes en Informix Geodetic DataBlade y Geodetic Extender (continuación)

Función de Informix Geodetic DataBlade	Función equivalente en DB2 Geodetic Extender	Comentarios para tipos de datos casi equivalentes
Unión	ST_Union	ST_Union tiene soporte para puntos y líneas además de polígonos

Tabla 17. Funciones de accesor correspondientes en Informix Geodetic DataBlade y Geodetic Extender

Función de Informix Geodetic DataBlade	Función equivalente en DB2 Geodetic Extender	Comentarios para tipos de datos casi equivalentes
Center	ST_MidPoint, ST_PointOnSurface	ST_MidPoint puede sustituir prácticamente a líneas. ST_PointOnSurface puede sustituir prácticamente a polígonos.
Coords	ST_PointN	
Dimension	ST_Dimension	
HasZValue	ST_Is3d	
IsGeoBox	Utilice la expresión IS OF o ST_GeometryType	
IsGeoCircle	Utilice la expresión IS OF o ST_GeometryType	
IsGeoEllipse	Utilice la expresión IS OF o ST_GeometryType	
IsGeoLineseg	Utilice la expresión IS OF o ST_GeometryType	
IsGeoPoint	Utilice la expresión IS OF o ST_GeometryType	
IsGeoPolygon	Utilice la expresión IS OF o ST_GeometryType	
IsGeoRing	Utilice la expresión IS OF o ST_GeometryType	
IsGeoString	Utilice la expresión IS OF o ST_GeometryType	
Latitude	ST_Y	
Longitude	ST_X	
NPoints	ST_NumPoints	
NRings	ST_NumGeometries, ST_NumInteriorRing	Utilice ST_NumGeometries para obtener el número total de anillos exteriores y sume ST_NumInteriorRings para cada polígono del conjunto de multipolígonos.
Ring	ST_GeometryN, ST_ExteriorRing, ST_InteriorRingN	Utilice ST_GeometryN junto con ST_ExteriorRing y ST_InteriorRingN
SRID	ST_SRID	
Zvalue	ST_Z	

Las siguientes funciones de accesor de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- IsLarge
- IsSmallArea

Tabla 18. Funciones de modificador equivalentes en Informix Geodetic DataBlade y Geodetic Extender

Función de Informix Geodetic DataBlade	Función equivalente en DB2 Geodetic Extender
SetSRID	ST_SRID

Las siguientes funciones de modificador de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- SetAltRange
- SetAltRangeZ
- SetDist
- SetTimeRange

Tabla 19. Funciones de medida equivalentes en Informix Geodetic DataBlade y Geodetic Extender

Función de Informix Geodetic DataBlade	Función equivalente en DB2 Geodetic Extender
Area	ST_Area
Distance	ST_Distance
Length	ST_Length, ST_Perimeter

La función de medida VoronoiResolution no tiene ninguna equivalencia en Geodetic Extender.

Tabla 20. Funciones de sentido descendente equivalentes en Informix Geodetic DataBlade y Geodetic Extender

Función de Informix Geodetic DataBlade	Función equivalente en DB2 Geodetic Extender
GeoBox	Utilice la expresión SQL TREAT
GeoCircle	Utilice la expresión SQL TREAT
GeoEllipse	Utilice la expresión SQL TREAT
GeoLineseg	Utilice la expresión SQL TREAT
GeoPoint	Utilice la expresión SQL TREAT
GeoPolygon	Utilice la expresión SQL TREAT
GeoRing	Utilice la expresión SQL TREAT
GeoString	Utilice la expresión SQL TREAT

Tabla 21. Funciones de constructor equivalentes en Informix Geodetic DataBlade y Geodetic Extender

Función de Informix Geodetic DataBlade	Función equivalente en DB2 Geodetic Extender
GeoCoords	ST_Point
GeoPoint	ST_Point

Las siguientes funciones de constructor de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- GeoBox
- GeoCircle
- GeoEllipse
- GeoLineseg

Tabla 22. Funciones de diagnóstico equivalentes en Informix Geodetic DataBlade y Geodetic Extender

Función de Informix Geodetic DataBlade	Función equivalente en DB2 Geodetic Extender
GeoTraceLevel	Recurso de rastreo de DB2
IsValidGeometry	ST_IsValid

Las siguientes funciones de diagnóstico de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- GeoInRowSize
- GeoOutOfRowSize
- GeoRelease
- GeoTotalSize
- GeoTraceLevelSet
- GeoWarningLevel
- GeoWarningLevelSet
- IsValidSDTS

Tabla 23. Tablas de catálogo del sistema equivalentes en Informix Geodetic DataBlade y Geodetic Extender

Tabla de catálogo del sistema de Informix Geodetic DataBlade	Vista de catálogo correspondiente en DB2 Geodetic Extender
GeoLenUnit	DB2GSE.ST_UNITS_OF_MEASURE
GeoSpatialRef	DB2GSE.SPATIAL_REF_SYS

Las siguientes tablas de catálogo del sistema de Geodetic DataBlade no tienen ningún equivalente entre las tablas o vistas de Geodetic Extender:

- GeoEllipsoid
- GeoParam
- GeoVoronoi

Las siguientes funciones de parámetro definibles por el usuario de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- GeoParamSessionGet
- GeoParamSessionSet

Las siguientes funciones de AltRange de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- AltRange
- Bottom
- Contains
- Equal
- Inside
- Intersect
- IsAny
- Outside
- Top

Las siguientes funciones de TimeRange de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- Begin
- Contains
- End
- Equal
- IsAny
- Inside
- Intersect
- Outside
- TimeRange

Las siguientes funciones de elipse de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- Azimuth
- Coords
- Major
- Minor

Las siguientes funciones de círculo de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- Coords
- Radius

Las siguientes funciones aritméticas de ángulo de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- Divide
- Minus
- Negate
- Plus
- Times

Tabla 24. Productos con conectividad de cliente equivalente en Geodetic DataBlade y DB2 Geodetic Extender

Conectividad de cliente de Informix Geodetic DataBlade	Conectividad de cliente equivalente en DB2 Geodetic Extender
ESQLC	SQC
ODBC	ODBC
JDBC	JDBC

Las siguientes conectividades de cliente de Geodetic DataBlade no tienen ninguna equivalencia en Geodetic Extender:

- Java API
- LIBMI

Llenado de columnas espaciales con datos geodésicos

Después de crear columnas espaciales y registrar aquellas en las que piensa crear un índice espacial, está preparado para llenar las columnas con datos geodésicos. Puede suministrar los datos geodésicos de las siguientes maneras:

- Importando los siguientes formatos de datos en una tabla nueva o existente:
  - Forma
  - SDE
- Insertando o actualizando valores en los siguientes formatos de datos:
  - Forma
  - SDE
  - Texto convencional (WKT)
  - Binario convencional (WKB)
  - GML (Geography Markup Language)

Restricciones:

- Con Spatial Extender Versión 8.2 no puede utilizar los mandatos del geocodificador ni procedimientos almacenados para convertir datos en datos geodésicos.
- Para el comportamiento geodésico, utilice sistemas de referencia espacial que tengan SRID comprendidos entre 2,000,000,000 y 2,000,001,000. Para obtener más información, consulte el apartado “Sistemas de referencia espacial” en la página 68.
- Los datos de formas y los datos de transferencia SDE deben estar en un sistema de coordenadas geográficas. Para obtener más información, consulte el apartado “Sistema de coordenadas geográficas” en la página 60.

Procedimiento:

El procedimiento a seguir para importar datos geodésicos es el mismo que se utiliza con los datos espaciales. Para obtener detalles, consulte el apartado “Importación de datos de formas a una tabla nueva o existente” en la página 90 e “Importación de datos de transferencia SDE a una tabla nueva o existente” en la página 92.

Conceptos relacionados:

- “Sistemas de referencia espacial” en la página 68

- "Sistema de coordenadas geográficas" en la página 60

Tareas relacionadas:

- "Importación de datos de formas a una tabla nueva o existente" en la página 90
- "Importación de datos de transferencia SDE a una tabla nueva o existente" en la página 92
- "Registro de columnas espaciales" en la página 87

Capítulo 18. Índices geodésicos

Puede crear índices Voronoi geodésicos que mejoren el rendimiento de las consultas de datos geodésicos. Este capítulo:

- Describe los índices Voronoi geodésicos
- Describes las estructuras de celdas Voronoi e indica cuándo puede seleccionar una estructura alternativa.
- Explica el modo de crear un índice Voronoi geodésico.

Índices Voronoi geodésicos

DB2® Geodetic Extender proporciona un índice Voronoi que agiliza el acceso a los datos geodésicos. Este índice organiza el acceso a los datos geodésicos utilizando formaciones teseladas Voronoi de la superficie de la tierra. Para obtener más información, consulte el apartado "Estructuras de celdas Voronoi" en la página 184.

Geodetic Extender calcula el círculo delimitador mínimo (MBC) para cada geometría. El MBC es un círculo que rodea una geometría geodésica. El índice Voronoi utiliza esta información de MBC para organizar los datos en una estructura de celdas. Una búsqueda que utiliza un índice Voronoi puede descender rápidamente por los datos organizados para encontrar objetos en el área de interés general y, a continuación, realizar pruebas más exactas en los propios objetos. Un índice Voronoi puede mejorar el rendimiento porque elimina la necesidad de examinar objetos situados fuera del área de interés. Sin un índice Voronoi, una consulta tendría que evaluar cada objeto para encontrar aquellos que coincidan con los criterios de búsqueda.

El optimizador considera que un índice Voronoi lo utilizarán todas las consultas que contienen las siguientes funciones en su cláusula WHERE:

- EnvelopesIntersect
- ST_Contains
- ST_Distance
- ST_EnvIntersects
- ST_Intersects
- ST_MBRIntersects
- ST_Within

Para obtener más información, consulte el apartado "Funciones que utilizan índices para optimizar consultas" en la página 128.

Cuando cree un índice Voronoi geodésico, puede elegir una estructura de celdas Voronoi alternativa. Para obtener más detalles, consulte el apartado "Consideraciones sobre la selección de una estructura de celdas Voronoi alternativa" en la página 185.

Conceptos relacionados:

- "Estructuras de celdas Voronoi" en la página 184
- "Consideraciones sobre la selección de una estructura de celdas Voronoi alternativa" en la página 185

- "Índices reticulares espaciales" en la página 104

Tareas relacionadas:

- "Creación de índices Voronoi geodésicos" en la página 187

Información relacionada:

- "Estructuras de celdas Voronoi suministradas con DB2 Geodetic Extender" en la página 190
- "Funciones que utilizan índices para optimizar consultas" en la página 128

Estructuras de celdas Voronoi

Para realizar cálculos eficaces, DB2® Geodetic Extender subdivide la superficie de la tierra en celdas más pequeñas y fáciles de gestionar, parecidas a las de un panal de abejas. Esta subdivisión se llama *formación teselada Voronoi*, y la estructura de datos que la describe se llama *estructura de celdas Voronoi*. Una *formación teselada Voronoi* es una estructura de celdas en la que el interior de cada celda está compuesto por todos los puntos cercanos a un punto del entramado particular, más que a cualquier otro punto. Las celdas de una estructura de celdas Voronoi son *cáscaras convexas*. Una cáscara convexa formada por un conjunto de puntos es el conjunto convexo más pequeño que incluye los puntos (o el polígono más pequeño que define el "exterior" de un grupo de puntos). Las estructuras de celdas Voronoi tienden a ser polígonos de forma irregular; el número y la ubicación de las celdas pueden ajustarse para coincidir con la densidad y la ubicación de los datos espaciales.

Por ejemplo, una estructura de celdas Voronoi puede subdividir la tierra en polígonos en función de la población humana. Allá donde la población y los datos sean densos aparecerán polígonos pequeños. Allá donde la población sea escasa aparecerán polígonos grandes.

La Figura 21 en la página 185 muestra la estructura Voronoi que se basa en la densidad de población mundial. Geodetic Extender utiliza esta estructura de celdas en sus cálculos espaciales.

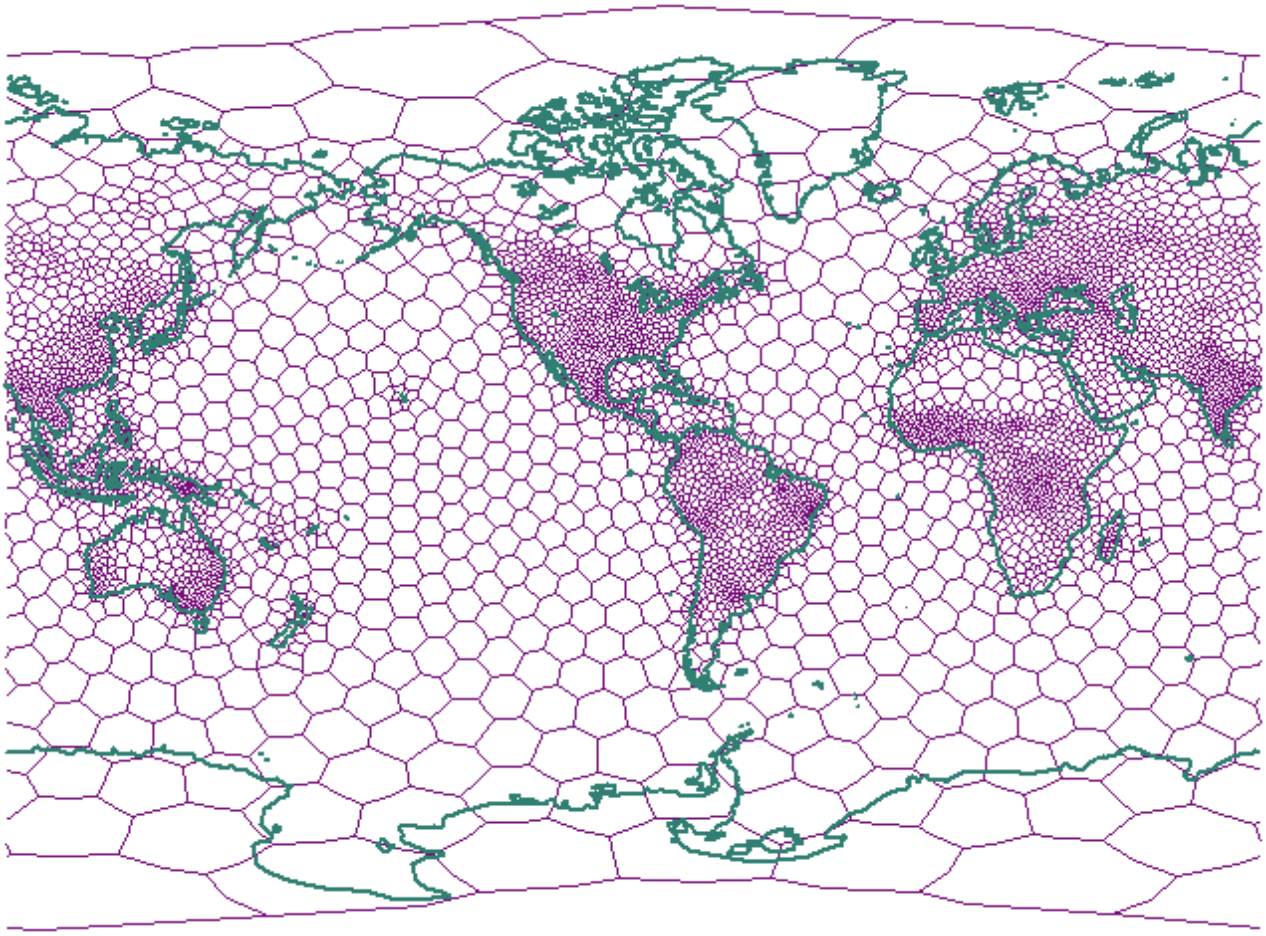


Figura 21. Estructura Voronoi basada en la densidad de población mundial

Conceptos relacionados:

- "Índices Voronoi geodésicos" en la página 183
- "Consideraciones sobre la selección de una estructura de celdas Voronoi alternativa" en la página 185

Tareas relacionadas:

- "Creación de índices Voronoi geodésicos" en la página 187

Información relacionada:

- "Sentencia CREATE INDEX para un índice Voronoi geodésico" en la página 188
- "Estructuras de celdas Voronoi suministradas con DB2 Geodetic Extender" en la página 190

Consideraciones sobre la selección de una estructura de celdas Voronoi alternativa

Todas las operaciones realizadas con geometrías geodésicas utilizan un ID de Voronoi de 1 que especifica la estructura de celdas Voronoi basada en la densidad de población mundial. Al crear un índice, si los datos están agrupados en una o varias áreas de la tierra, como los datos sobre calles de uno o varios países, puede elegir una estructura de celdas Voronoi alternativa que tiene celdas más pequeñas

en las áreas en que están ubicados los datos (porque la resolución es inversamente proporcional al tamaño de las celdas). DB2® Geodetic Extender proporciona varias estructuras de celdas Voronoi para indexar que podrían ajustarse perfectamente a sus datos. Para obtener una lista de las estructuras alternativas y diagramas disponibles que ilustran dichas estructuras de celdas, consulte el apartado "Estructuras de celdas Voronoi suministradas con DB2 Geodetic Extender" en la página 190.

Restricción: Sólo puede elegir una estructura de celdas Voronoi alternativa cuando cree un índice Voronoi geodésico.

La estructura dodeca04 (ID de Voronoi 12) es adecuada para datos que están distribuidos uniformemente sobre toda la superficie de la tierra, como las imágenes que envían los satélites. Todas las celdas son más o menos uniformes en cuanto al tamaño y la resolución, en el peor de los casos, es de 10 centímetros aproximadamente. Considere la posibilidad de utilizar una estructura de celdas Voronoi distinta de la estructura de la población mundial por omisión (ID de Voronoi 1) o de la estructura dodeca04 si alguna de las siguientes condiciones se aplica a sus datos o a su aplicación:

Alta resolución
Si necesita determinar si forman intersección los objetos separados por menos de 10 centímetros, deberá utilizar una estructura de celdas Voronoi que tenga celdas más pequeñas en las regiones donde se ubican sus datos. La resolución es inversamente proporcional al tamaño de las celdas.

Polígonos con muchos vértices
Si los datos están formados por polígonos que tienen un número relativamente grande de vértices y su superficie es relativamente pequeña, se recomienda utilizar en su lugar una estructura de celdas Voronoi que tenga más celdas en las regiones de interés. Si la mayoría de los polígonos tienen 50 vértices o menos, probablemente no deseará hacerlo. Si los únicos polígonos del conjunto de datos que tienen muchos vértices tienen tamaño de continente, probablemente tampoco deseará hacerlo.

Si tiene muchos polígonos de 3000 vértices con un tamaño como el de un condado de EE.UU., podrá mejorar significativamente el rendimiento de la consulta utilizando una estructura de celdas distinta, especialmente si la aplicación realiza varias consultas de polígono-forma intersección-polígono.

Datos muy densos
Si los datos se concentran en regiones muy pequeñas (por ejemplo, si tiene cientos de objetos por kilómetro cuadrado) , podrá mejorar el rendimiento de la consulta utilizando una estructura de celdas Voronoi cuya densidad de celdas coincida con la densidad de los datos.

Conceptos relacionados:

- "Índices Voronoi geodésicos" en la página 183
- "Estructuras de celdas Voronoi" en la página 184

Tareas relacionadas:

- "Creación de índices Voronoi geodésicos" en la página 187

Información relacionada:

- "Sentencia CREATE INDEX para un índice Voronoi geodésico" en la página 188
- "Estructuras de celdas Voronoi suministradas con DB2 Geodetic Extender" en la página 190

Creación de índices Voronoi geodésicos

DB2 Geodetic Extender proporciona un nuevo método de acceso espacial que permite crear índices en columnas que contengan datos geodésicos. Las consultas que utilizan un índice se pueden ejecutar más rápidamente.

Requisitos previos:

Para crear un índice Voronoi geodésico, su ID de usuario debe tener los mismos privilegios y autorizaciones que cuando crea un índice reticular espacial (consulte el apartado "Creación de índices reticulares espaciales" en la página 111).

Restricciones:

Las mismas restricciones que existen para crear índices mediante la sentencia CREATE INDEX son aplicables cuando crea un índice Voronoi geodésico. Es decir, la columna en la que crea un índice debe ser una columna de tabla base, no una columna de vista ni una columna de seudónimo. DB2 UDB resolverá los alias durante el proceso.

Procedimiento:

Puede crear un índice Voronoi geodésico de cualquiera de las maneras siguientes:

- Utilice la ventana Crear índice del Centro de control de DB2.
- Utilice la sentencia CREATE INDEX de SQL con la extensión db2gse.spatial_index en la cláusula EXTEND USING.

Para crear un índice Voronoi geodésico utilizando el Centro de control:

1. En el Centro de control, pulse con el botón derecho del ratón la tabla que tiene la columna espacial en la que desea para crear un índice Voronoi y seleccione **Spatial Extender ⟶ Índices espaciales** en el menú desplegable. Se abrirá la ventana Índices espaciales.
2. Siga las instrucciones de la ayuda en línea para la ventana Índices espaciales. Para visualizar esas instrucciones, pulse el botón **Ayuda** en la ventana Índices espaciales.

Para crear un índice Voronoi geodésico utilizando la sentencia CREATE INDEX de SQL: Emita la sentencia CREATE INDEX utilizando la cláusula EXTEND USING y la extensión de índice reticular db2gse.spatial_index.

Ejemplo:

En el siguiente ejemplo, la sentencia CREATE INDEX crea el índice geodésico STORESX1 en la columna espacial LOCATION de la tabla CUSTOMERS:

```
CREATE INDEX storesx1
   ON customers (location)
   EXTEND USING db2gse.spatial_index (-1, -1, 1)
```

Para un índice Voronoi geodésico debe especificar el valor –1 en los dos primeros parámetros de la cláusula USING db2gse.spatial_index. Para obtener detalles, consulte el apartado "Sentencia CREATE INDEX para un índice Voronoi geodésico" en la página 188.

Conceptos relacionados:

- "Índices Voronoi geodésicos" en la página 183

- "Estructuras de celdas Voronoi" en la página 184
- "Consideraciones sobre la selección de una estructura de celdas Voronoi alternativa" en la página 185

Tareas relacionadas:

- "Creación de índices reticulares espaciales" en la página 111

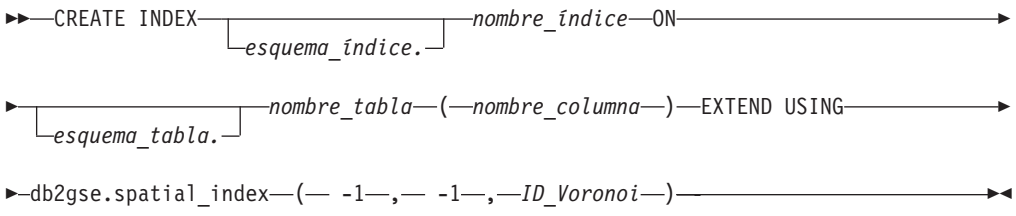
Información relacionada:

- "Sentencia CREATE INDEX para un índice Voronoi geodésico" en la página 188
- "Estructuras de celdas Voronoi suministradas con DB2 Geodetic Extender" en la página 190
- "Funciones que utilizan índices para optimizar consultas" en la página 128

Sentencia CREATE INDEX para un índice Voronoi geodésico

Utilice la sentencia CREATE INDEX con la cláusula EXTEND USING para crear un índice Voronoi geodésico.

Sintaxis:



Donde:

esquema_índice
: Nombre del esquema al que debe pertenecer el índice que está creando. Si no especifica un nombre, DB2 UDB utiliza el nombre de esquema que está almacenado en el registro especial CURRENT SCHEMA.

nombre_índice
: Nombre, no calificado, del índice geodésico que está creando.

esquema_tabla
: Nombre del esquema al que pertenece la tabla donde reside *nombre_columna*. Si no especifica un nombre, DB2 UDB utiliza el nombre de esquema que está almacenado en el registro especial CURRENT SCHEMA.

nombre_tabla
: Nombre, no calificado, de la tabla donde reside *nombre_columna*.

nombre_columna
: Nombre de la columna espacial en la que se ha creado el índice Voronoi geodésico.

ID_Voronoi
: Un número entero que identifica el ID de la estructura de celdas Voronoi. Se encuentran disponibles catorce estructuras de celdas Voronoi. Un ID de Voronoi de 1 especifica la estructura de celdas Voronoi basada en la densidad de población mundial que DB2 Geodetic Extender también utiliza para todas las operaciones espaciales.

Ejemplos:

En el siguiente ejemplo, la sentencia CREATE INDEX crea el índice geodésico STORESX1 en la columna espacial LOCATION de la tabla CUSTOMERS:

```
CREATE INDEX storesx1
   ON customers (location)
   EXTEND USING db2gse.spatial_index (-1, -1, 1)
```

El optimizador considera que un índice Voronoi lo utilizarán todas las consultas que contienen las siguientes funciones en su cláusula WHERE:

- ST_Contains
- ST_Distance
- ST_Intersects
- ST_MBRIntersects
- ST_EnvIntersects
- EnvelopesIntersect
- ST_Within

Las siguientes sentencias muestran el uso de un índice Voronoi. En primer lugar, inserte datos en la tabla CUSTOMER. Puede entrar los valores directamente, tal como se muestra en esta primera sentencia INSERT:

```
INSERT INTO customer
(id, last_name, first_name, address, city, state, zip,
location)
VALUES
('123-456789', 'Duck', 'Donald',
'123 Mallard Way', 'Wetland Marsh', 'ND', '55555-5555',
db2gse.ST_GeomFromWKT('POINT(123.123, 45.67)', 2000000000))
```

De forma alternativa, puede utilizar variables en una aplicación, tal como muestra la siguiente consulta, para insertar valores en una tabla:

```
INSERT INTO customer
(id, last_name, first_name,
address, city, state, zip,
location)
VALUES
(:mid, :mlast, :mfirst,
:maddress, :mcity, :mstate, :mzip,
db2gse.ST_GeomFromWKB(:mlocation))
```

La siguiente sentencia UPDATE modifica los datos insertados. No utiliza el índice STORESX1 porque no utiliza las funciones ST_Contains, ST_Distance, ST_Intersects, ST_MBRIntersects, ST_EnvIntersects, EnvelopesIntersect ni ST_Within en su cláusula WHERE.

```
UPDATE customer
SET location = db2gse.ST_GeomFromWKT('POINT(123.123, 45.67)',
2000000000)
WHERE id = '123-456789';
```

Las siguientes sentencias DELETE pueden utilizar el índice STORESX1, si el optimizador determina que el índice mejora el rendimiento, porque las sentencias DELETE utilizan la función ST_Within y las funciones ST_Intersects en sus cláusulas WHERE respectivamente:

```
DELETE FROM customers
WHERE db2gse.ST_Within(location, :BayArea) = 1;
```

```
DELETE FROM customers
WHERE db2gse.ST_Intersects(c.location, :BayArea) = 1
```

Las dos sentencias SELECT siguientes también pueden utilizar el índice STORESX1:

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

Conceptos relacionados:

- "Índices Voronoi geodésicos" en la página 183
- "Estructuras de celdas Voronoi" en la página 184
- "Consideraciones sobre la selección de una estructura de celdas Voronoi alternativa" en la página 185

Tareas relacionadas:

- "Creación de índices Voronoi geodésicos" en la página 187

Información relacionada:

- "Estructuras de celdas Voronoi suministradas con DB2 Geodetic Extender" en la página 190

Estructuras de celdas Voronoi suministradas con DB2 Geodetic Extender

Cada estructura de celdas Voronoi cubre toda la tierra. En las ilustraciones siguientes, sólo se muestran aquellas porciones de la tierra contenidas en el área en que las celdas son densas para dicha estructura de celdas Voronoi. Cuando seleccione una estructura de celdas Voronoi, tenga en cuenta que las celdas situadas fuera de las áreas ilustradas serán grandes y su resolución será, como corresponde, inferior. Si los datos se encuentran en dichas áreas escasas, es posible que se reduzca el rendimiento de la consulta.

La tabla siguiente lista las estructuras de celdas Voronoi que suministra DB2 Geodetic Extender. Dichas estructuras de celdas Voronoi las proporciona Geodyssey Ltd.

Tabla 25. Estructuras de celdas Voronoi

Descripción	ID de Voronoi	Ilustración
Mundo, basada en la densidad de población	1	Figura 22 en la página 191
Estados Unidos	2	Figura 23 en la página 192
Canadá	3	Figura 24 en la página 193
India	4	Figura 25 en la página 194
Japón	5	Figura 26 en la página 195
África	6	Figura 27 en la página 196
Australia	7	Figura 28 en la página 197
Europa	8	Figura 29 en la página 198
Norteamérica	9	Figura 30 en la página 199
Sudamérica	10	Figura 31 en la página 200

Tabla 25. Estructuras de celdas Voronoi (continuación)

Descripción	ID de Voronoi	Ilustración
Mediterráneo	11	Figura 32 en la página 201
Mundo, distribución de datos uniforme, resolución media (dodeca04)	12	Figura 33 en la página 202
Mundo, basada en la producción industrial (naciones del G7)	13	Figura 34 en la página 203
Mundo, distribución de datos uniforme, baja resolución (isotipo)	14	Figura 35 en la página 204

Mundo, basada en la densidad de población (ID de Voronoi: 1)

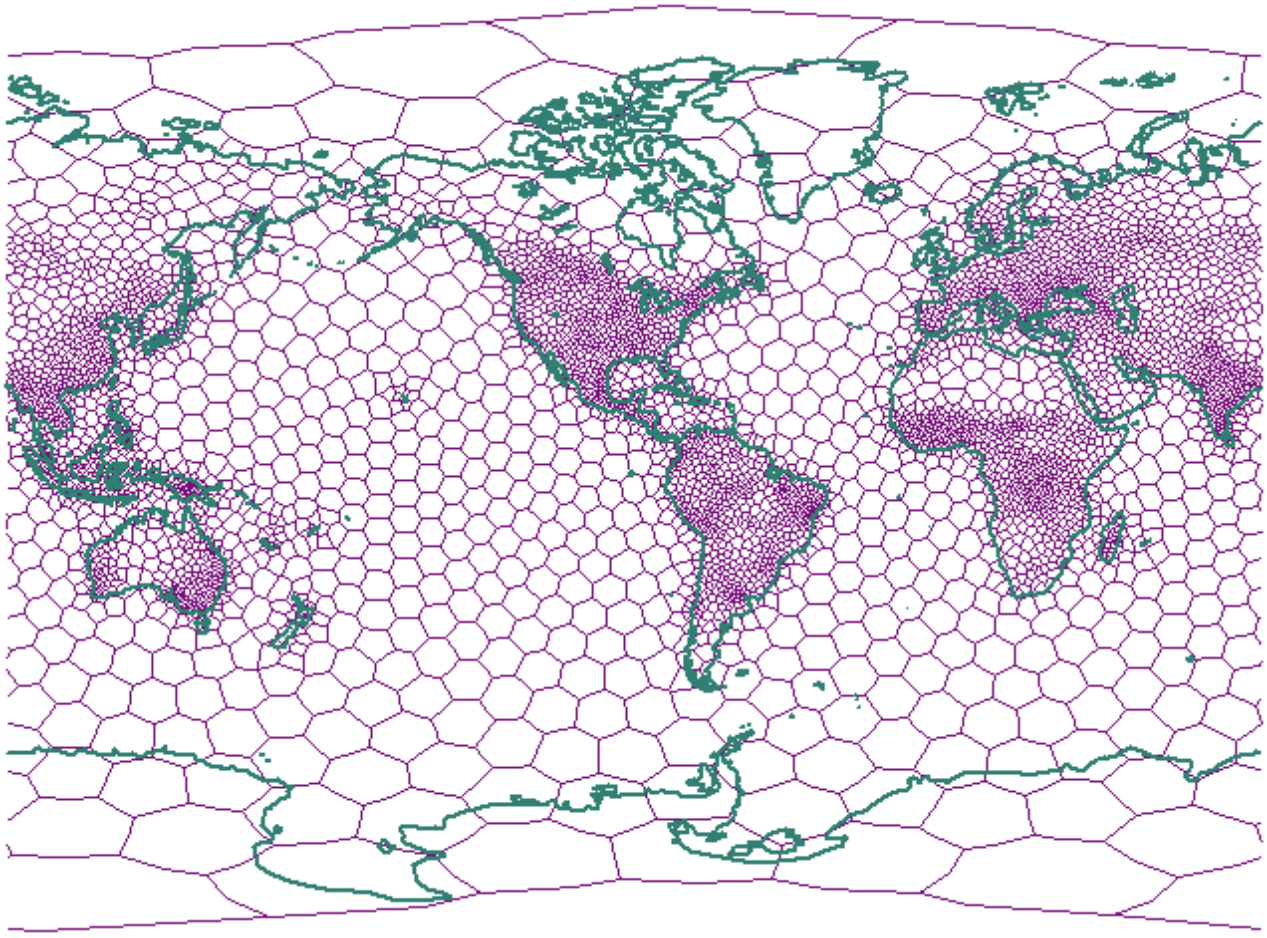


Figura 22. Estructura de celdas Voronoi para el mundo (población)

Estados Unidos (ID de Voronoi: 2)

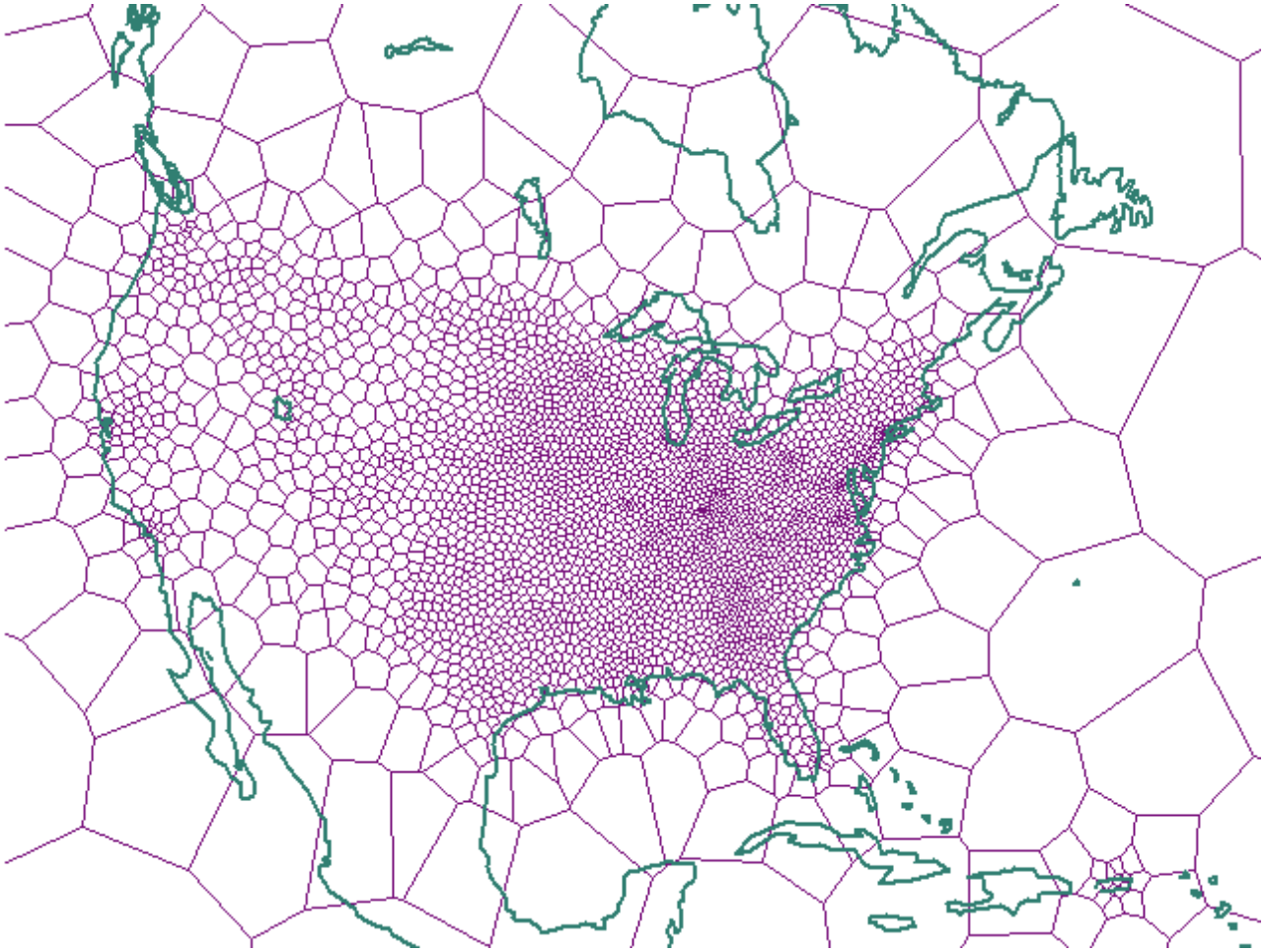


Figura 23. Estructura de celdas Voronoi para los EE.UU.

Canadá (ID de Voronoi: 3)

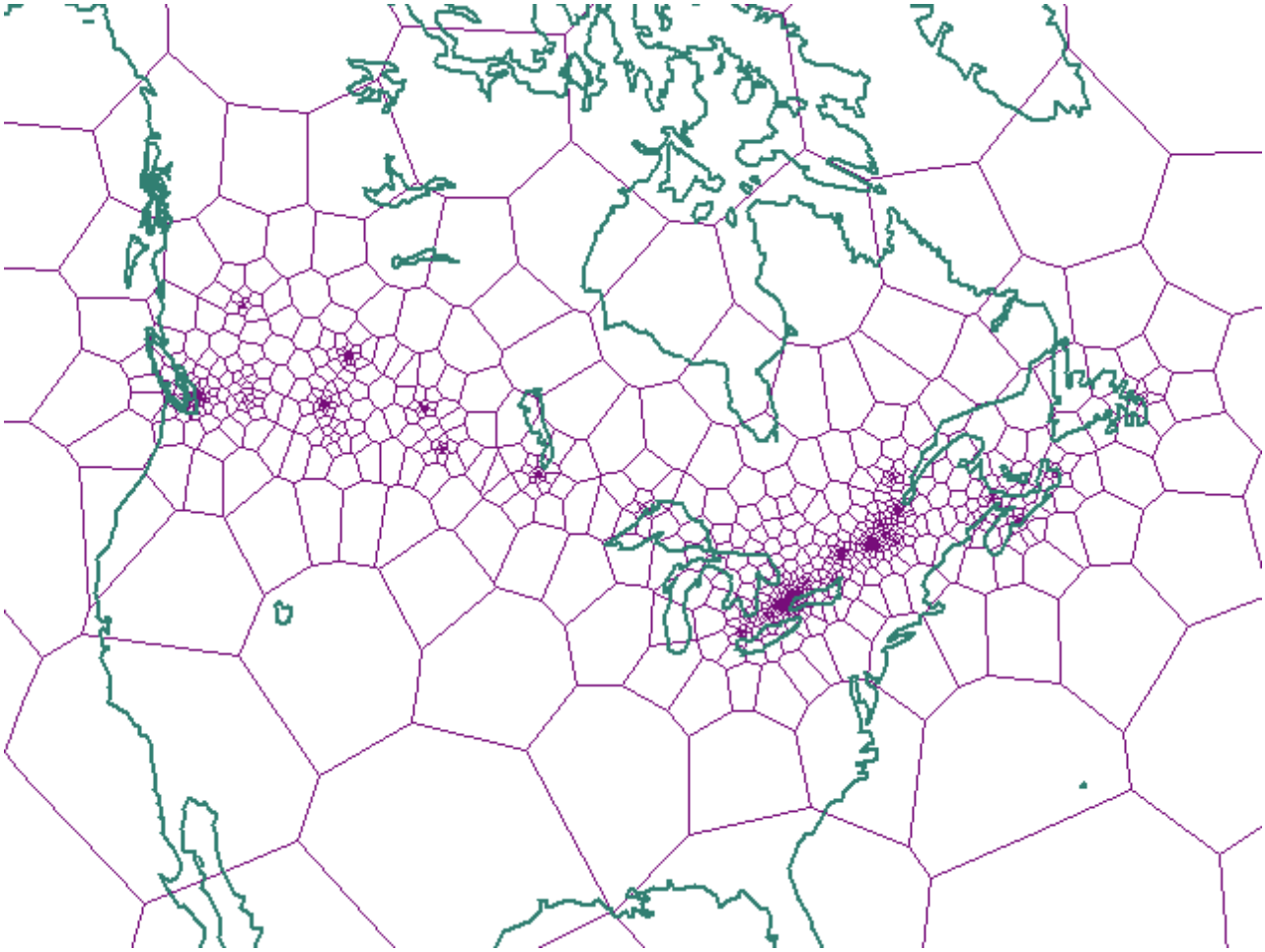


Figura 24. Estructura de celdas Voronoi para Canadá

India (ID de Voronoi: 4)

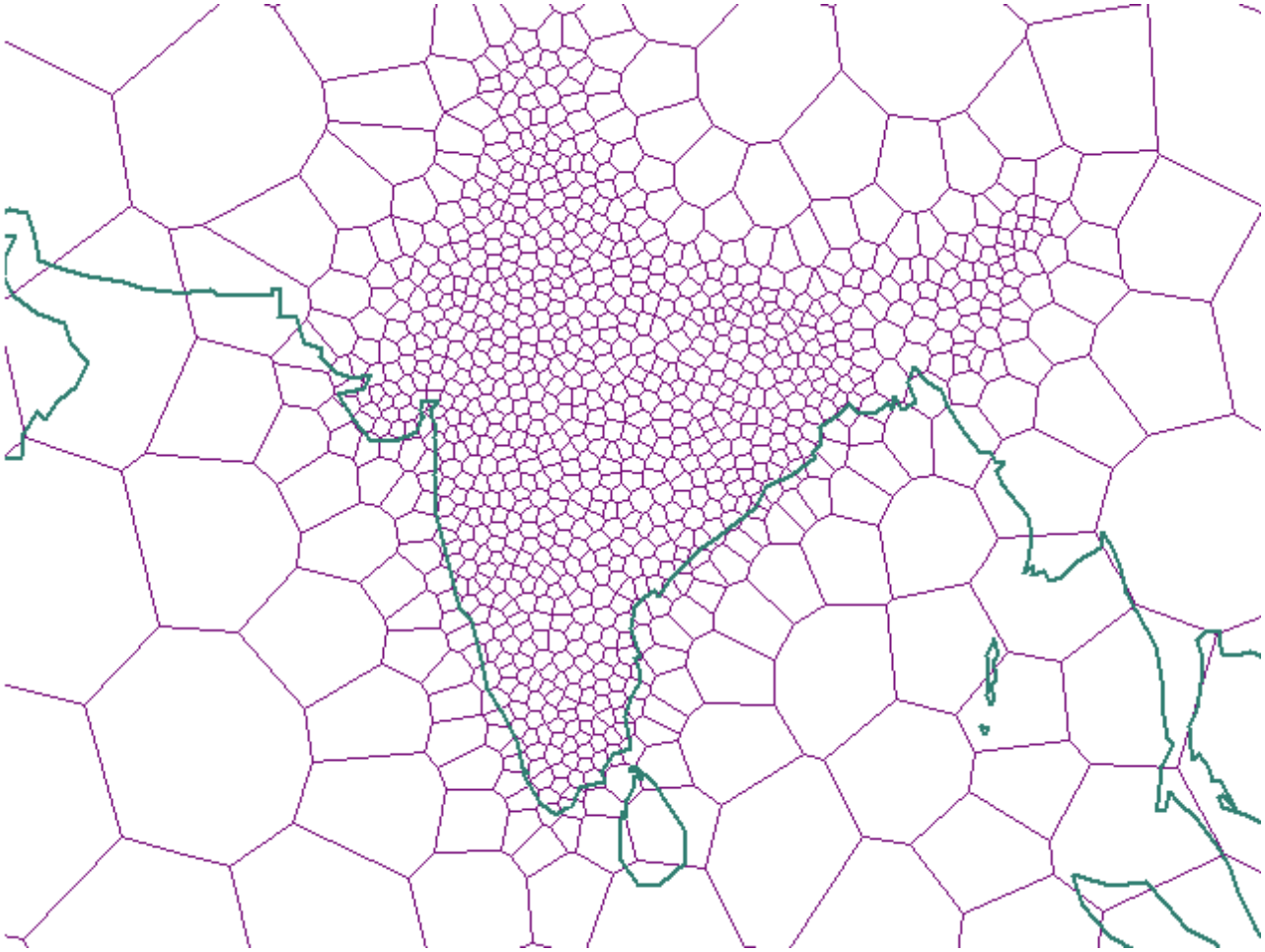


Figura 25. Estructura de celdas Voronoi para la India

Japón (ID de Voronoi: 5)

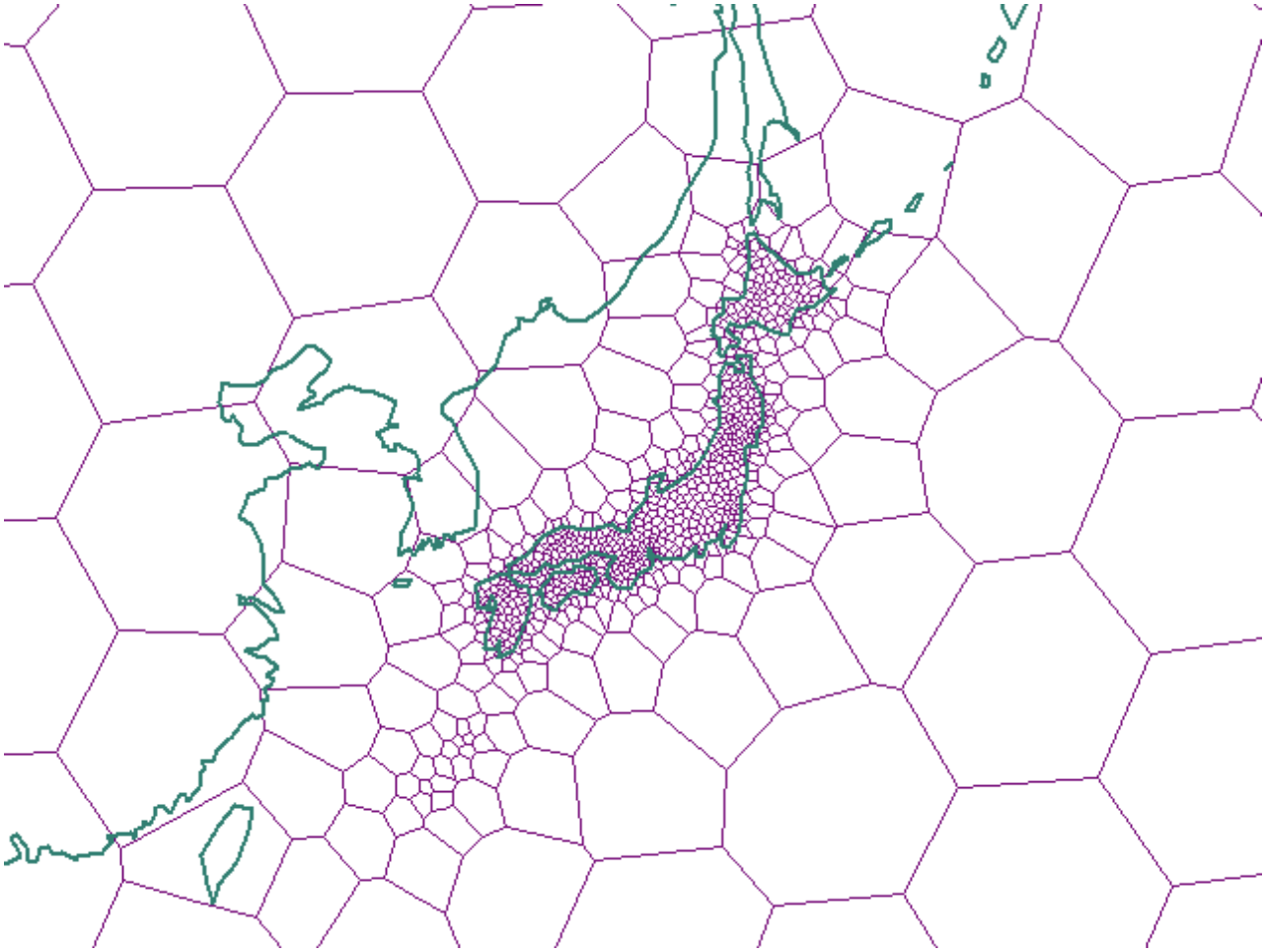


Figura 26. Estructura de celdas Voronoi para Japón

África (ID de Voronoi: 6)

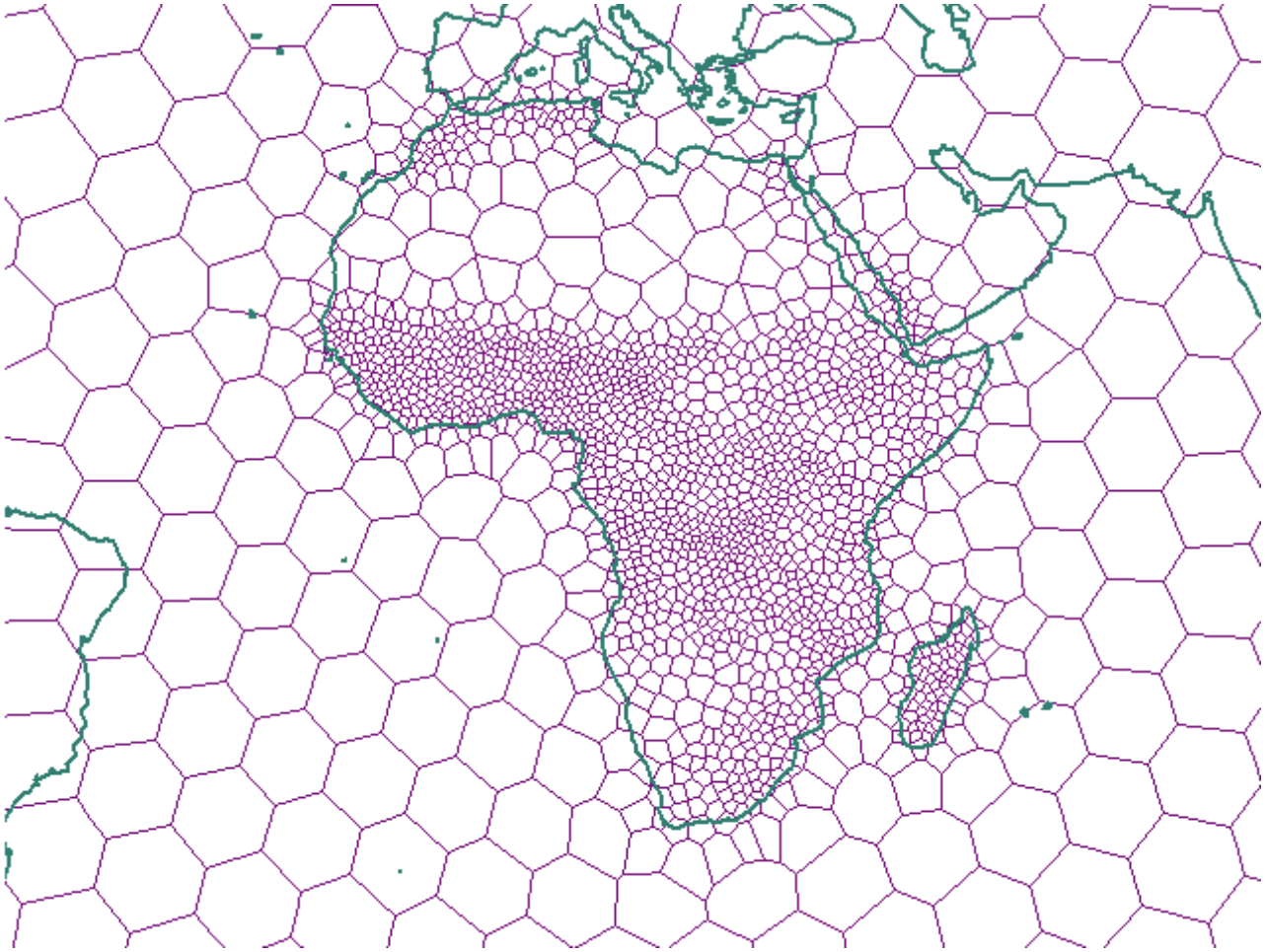


Figura 27. Estructura de celdas Voronoi para África

Australia (ID de Voronoi: 7)

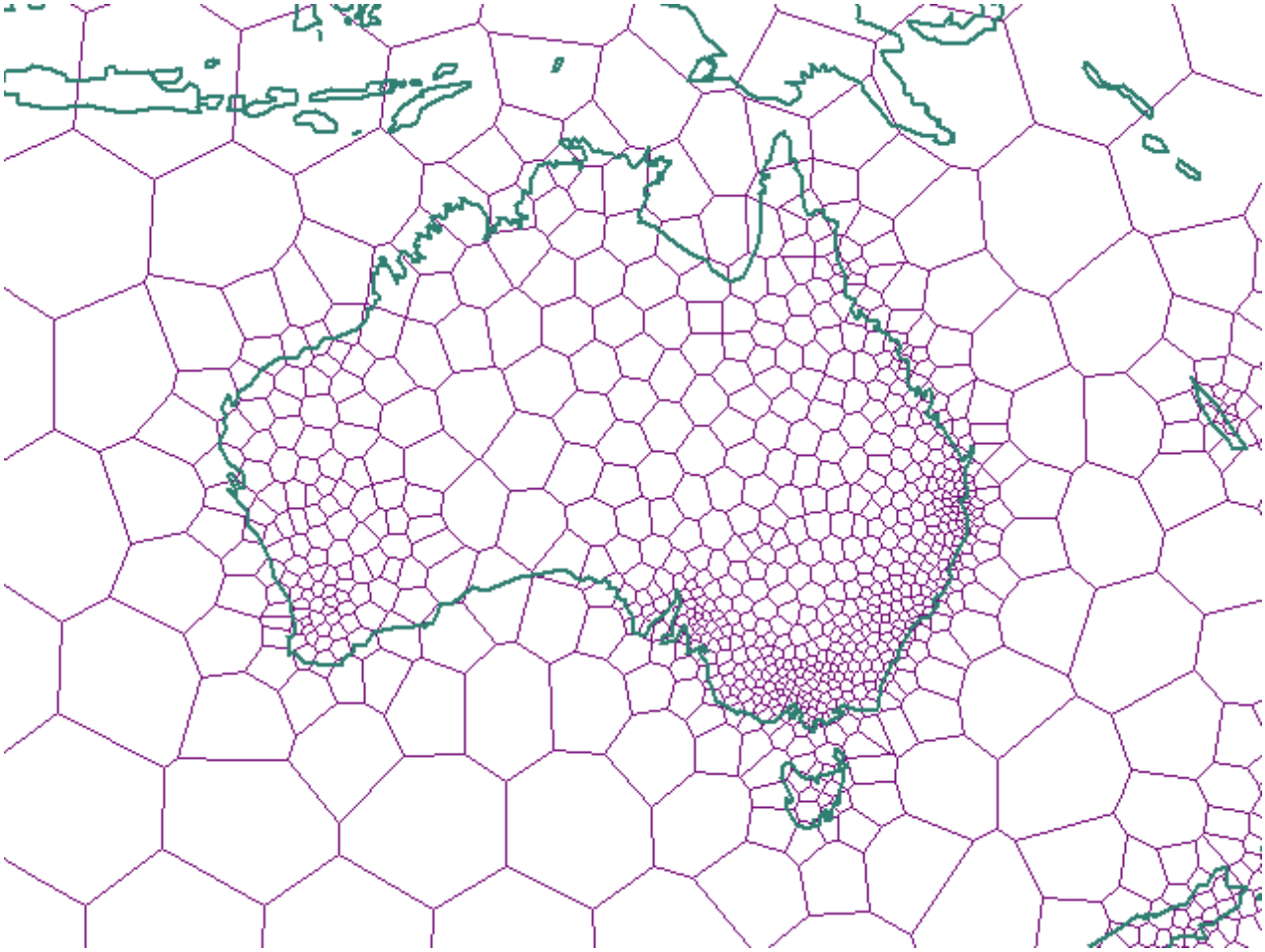


Figura 28. Estructura de celdas Voronoi para Australia

Europa (ID de Voronoi: 8)

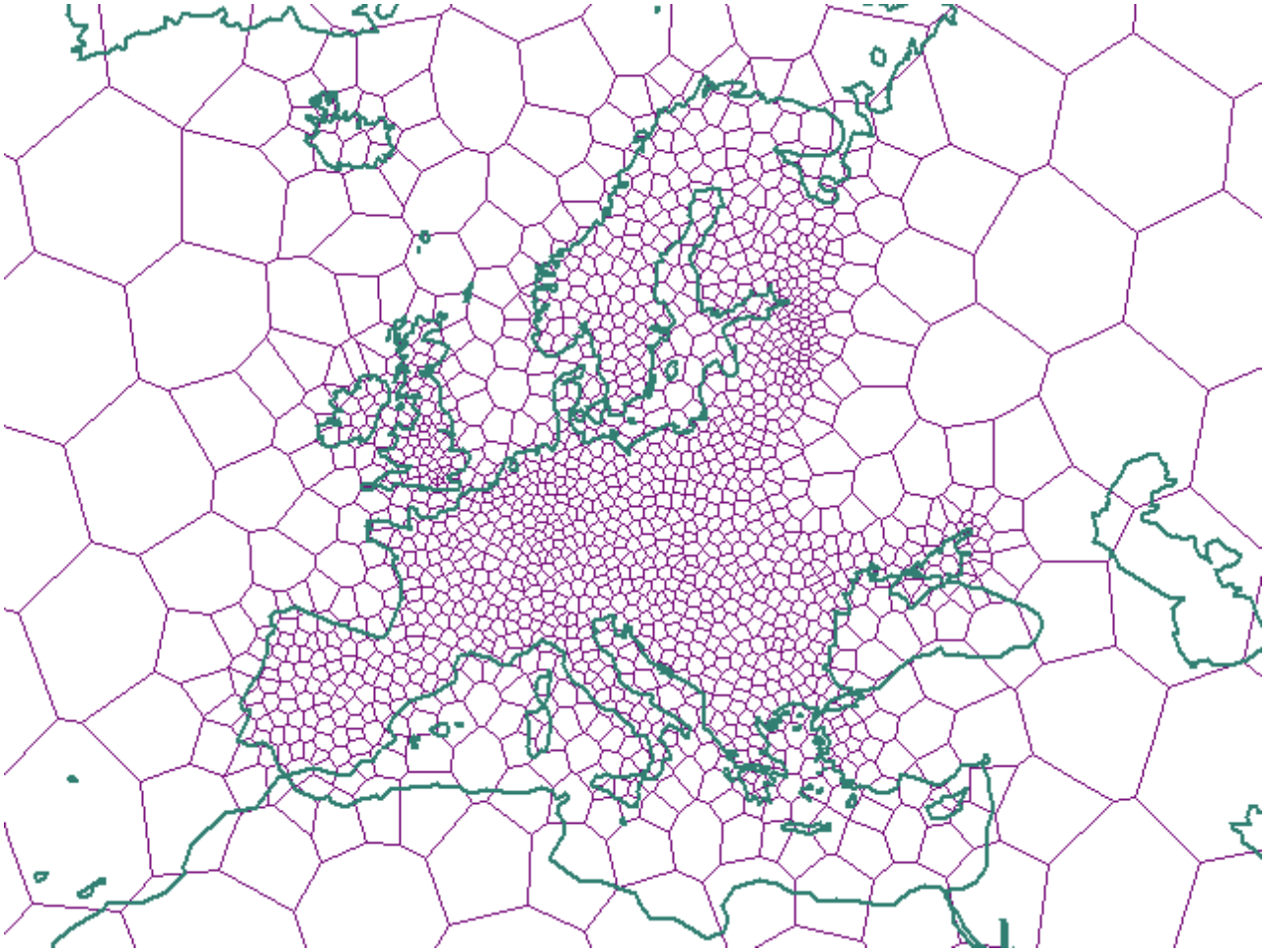


Figura 29. Estructura de celdas Voronoi para Europa

Norteamérica (ID de Voronoi: 9)

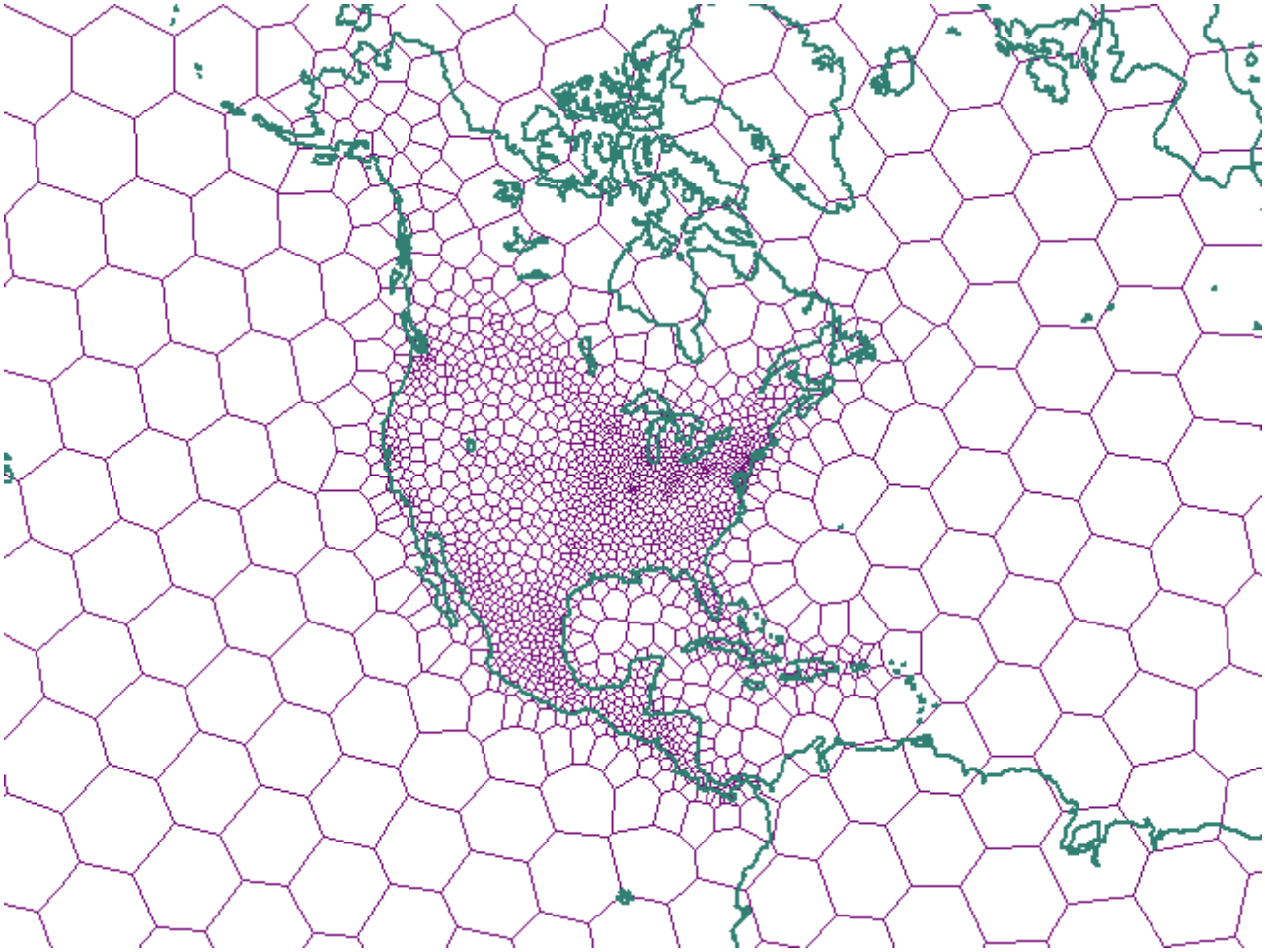


Figura 30. Estructura de celdas Voronoi para Norteamérica

Sudamérica (ID de Voronoi: 10)

Figura 31. Estructura de celdas Voronoi para Sudamérica

Mediterráneo (ID de Voronoi: 11)

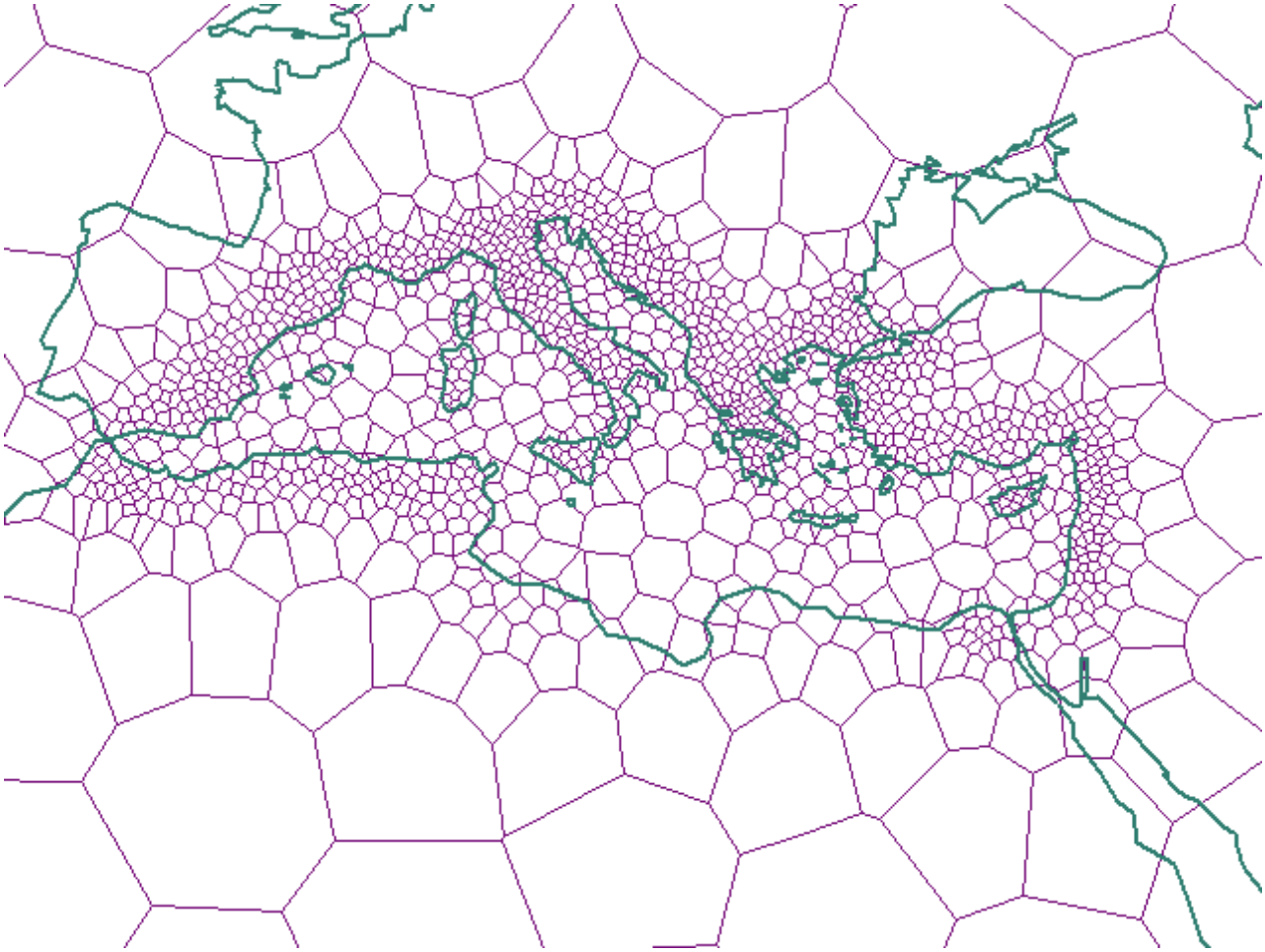


Figura 32. Estructura de celdas Voronoi para el área mediterránea

Mundo, distribución de datos uniforme, resolución media – dodeca04 (ID de Voronoi: 12)

Figura 33. Estructura de celdas Voronoi para el mundo (dodeca04)

Mundo, países industrializados – países del G7 (ID de Voronoi: 13)

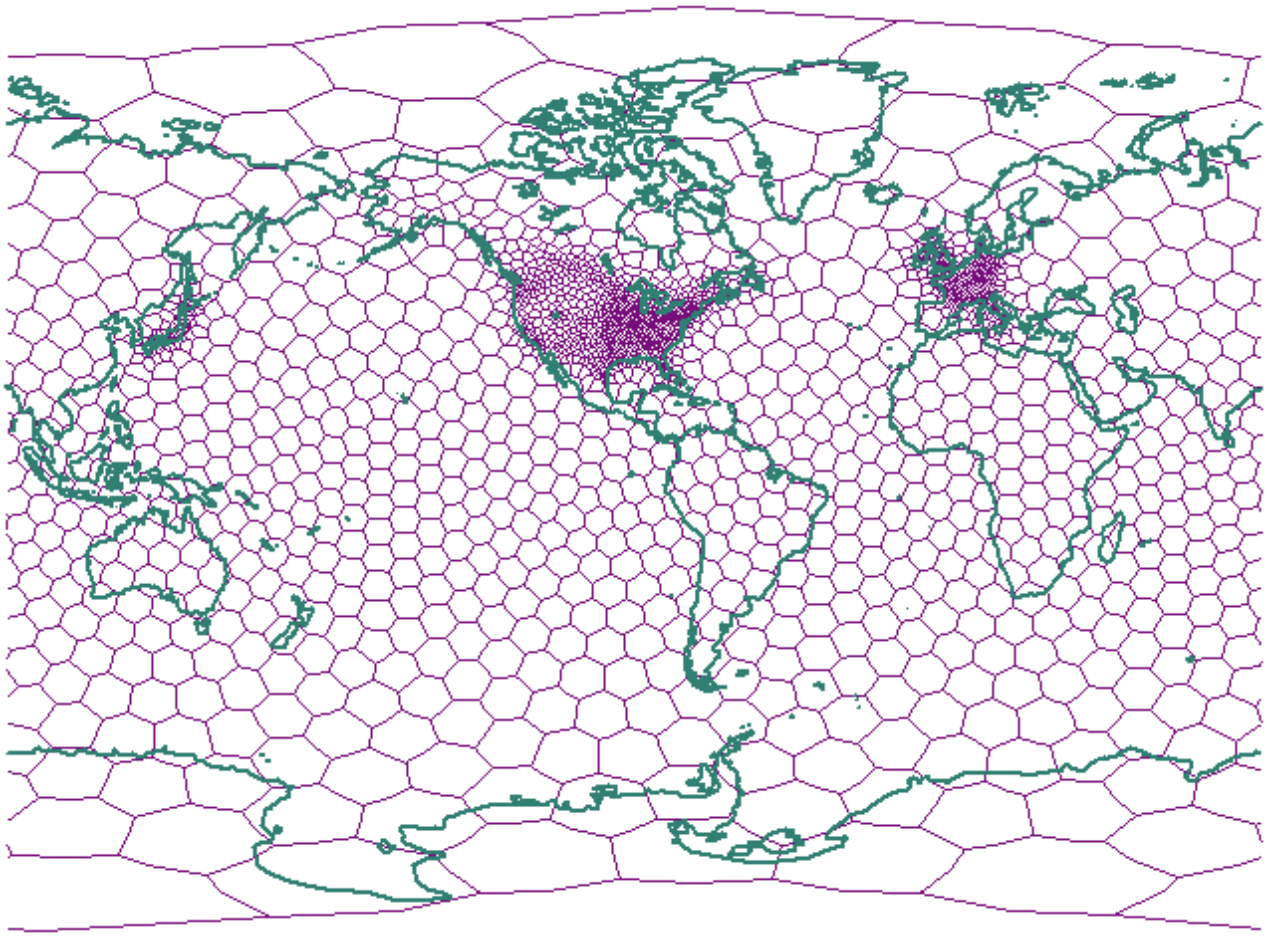


Figura 34. Estructura de celdas Voronoi para (naciones del g7)

Mundo, distribución de datos uniforme, baja resolución – isotipo (ID de Voronoi: 14)

Figura 35. Estructura de celdas Voronoi para el mundo (isotipo)

Capítulo 19. Diferencias en la utilización de datos geodésicos y espaciales

Este capítulo describe las siguientes diferencias en la utilización de datos geodésicos y espaciales:

- Atributos x e y mínimos y máximos para tipos de datos ST_Geometry
- Diferencias entre la utilización de representaciones terrestres y planas
- Funciones espaciales soportadas por DB2 Geodetic Extender y diferencias en el comportamiento de las funciones
- Procedimientos almacenados y vistas de catálogo soportados por DB2 Geodetic Extender
- Sistemas de referencia espacial geodésicos (datums) adicionales y elipsoides geodésicos

Atributos x e y mínimos y máximos

DB2® Geodetic Extender utiliza un círculo delimitador mínimo (MBC) en lugar de un rectángulo delimitador mínimo para organizar los datos en estructuras de celdas para el índice Voronoi geodésico.

En las geometrías geodésicas, el MBC es un círculo que rodea las geometrías, y los atributos x e y mínimos y máximos tienen los siguientes valores internos:

xmin El término *i* del coseno de dirección del centro del círculo delimitador.

xmax El término *j* del coseno de dirección del centro del círculo delimitador.

ymin El término *k* del coseno de dirección del centro del círculo delimitador.

ymax El *radio_arco* del círculo delimitador.

En las geometrías geodésicas, las funciones ST_MinX, ST_MaxX, ST_MinY y ST_MaxY muestran puntos a lo largo del MBC. Los resultados de estas funciones todavía generan valores de longitud y latitud similares a las geometrías espaciales, pero los resultados pueden diferir en las geometrías geodésicas de la siguiente manera:

- Si el MBC cruza la línea de datos, el valor de ST_MinX es mayor que el valor de ST_MaxX. Por ejemplo, si el centro de un MBC se encuentra en la línea de datos y tiene un radio de 5 grados, el valor de ST_MinX es 175 y el valor de ST_MaxX es –175.
- Si el MBC incluye el polo norte o el polo sur, ST_MinX es –180 y ST_MaxX es 180.
- Si el MBC incluye el polo norte, el valor de ST_MaxY es 90.
- Si el MBC incluye el polo sur, el valor de ST_MinY es –90.

Diferencias de utilización entre las representaciones terrestres planas y redondas

DB2 Spatial Extender y DB2 Geodetic Extender utilizan distintas tecnologías básicas:

- Spatial Extender utiliza un mapa plano (o planar) basado en coordenadas proyectadas. Sin embargo, la proyección en un mapa no es capaz de representar fielmente toda la tierra porque los mapas tienen bordes y la tierra carece de ellos.
- Geodetic Extender utiliza un elipsoide como modelo y considera que la tierra es un globo sin costuras que no tiene distorsiones en los polos ni en los bordes en el meridiano 180º.

En este apartado, el término ″tierra plana″ alude al uso de una proyección para representar toda la tierra. El término ″tierra redonda″ alude al uso de un sistema de referencia que utiliza un elipsoide como modelo de la tierra.

Las distintas tecnologías producen diferencias en el modo en que las geometrías se manejan bajo determinadas situaciones, especialmente las que se ilustran en este tema:

- Segmentos lineales (y distancias medidas) que cruzan el meridiano 180º.
- Polígonos que ocupan el meridiano 180º.
- Rectángulos delimitadores mínimos que cruzan el meridiano 180º.
- Polígonos que delimitan un polo.
- Polígonos que representan hemisferios, cinturones ecuatoriales o toda la tierra.

Geodetic Extender ofrece ventajas exclusivas al trabajar con geometrías que cruzan el meridiano 180º o que están cercanas a un polo, donde la representación terrestre plana que utiliza Spatial Extender encuentra limitaciones.

Segmentos lineales que cruzan el meridiano 180º

La Figura 36 en la página 207 muestra las distintas formas en que Spatial Extender y Geodetic Extender manejan un segmento lineal que cruza el meridiano 180º. En este ejemplo, se utiliza el segmento lineal para calcular la distancia entre Anchorage y Tokio. Geodetic Extender calcula la distancia entre dos puntos a lo largo de una geodésica, la trayectoria más corta entre dos puntos de la elipsoide (consultar el apartado “Distancias geodésicas” en la página 168). Los dos puntos pueden estar ubicados en cualquier lugar del globo, y Geodetic Extender elige correctamente un segmento lineal que transcurre en dirección oeste desde Anchorage hasta Tokio porque utiliza la representación terrestre redonda. Sin embargo, puesto que Spatial Extender utiliza una proyección de mapa plano, Spatial Extender ignora que un segmento lineal podría conectar Anchorage y Tokio de esa manera, por lo que elige un segmento lineal mucho más largo que transcurre en dirección este hasta Tokio. La proyección de mapa plano tiene el meridiano –180º en el borde izquierdo y el meridiano 180º en el borde derecho.

Para obtener un resultado correcto utilizando Spatial Extender deberá realizar una de las siguientes acciones:

- Dividir el segmento lineal en dos segmentos lineales, uno al este del meridiano 180º y el otro al oeste del mismo.
- Volver a proyectar los datos de modo que el meridiano 180º no esté en un borde.

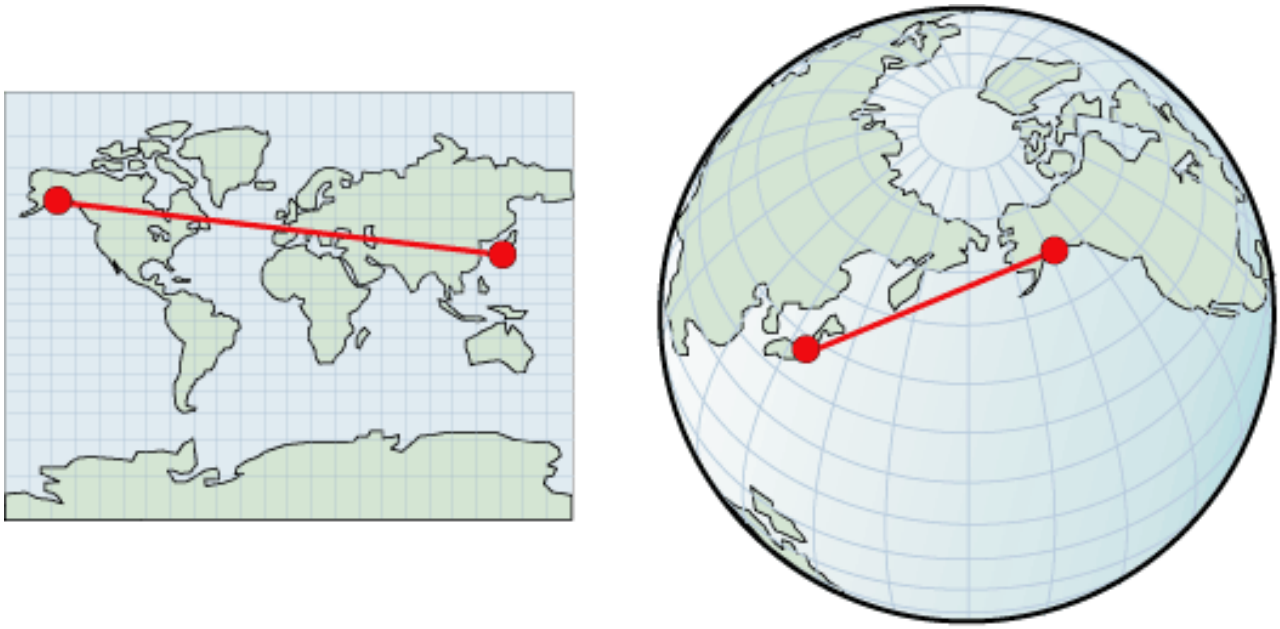


Figura 36. Líneas que cruzan el meridiano 180º

Polígonos que ocupan el meridiano 180º

Para trabajar con un polígono que ocupa el meridiano 180º, la representación terrestre plana (Spatial Extender) necesita que el usuario divida el polígono en dos partes, un polígono para la porción que se encuentra al este del meridiano 180º y otro polígono para la porción que se encuentra al oeste del meridiano:

```
  MULTIPOLYGON(
((–180 30, –165 30, –165 40, –180 40, –180 30)),
((180 30, 180 40, 165 40, 165 30, 180 30)))
```

Como muestra la Figura 37 en la página 208, la representación terrestre redonda (Geodetic Extender) no requiere dicha división y puede utilizar un único polígono sin modificar:

```
  POLYGON((165 30, –165 30, –165 40, 165 40, 165 30))
```

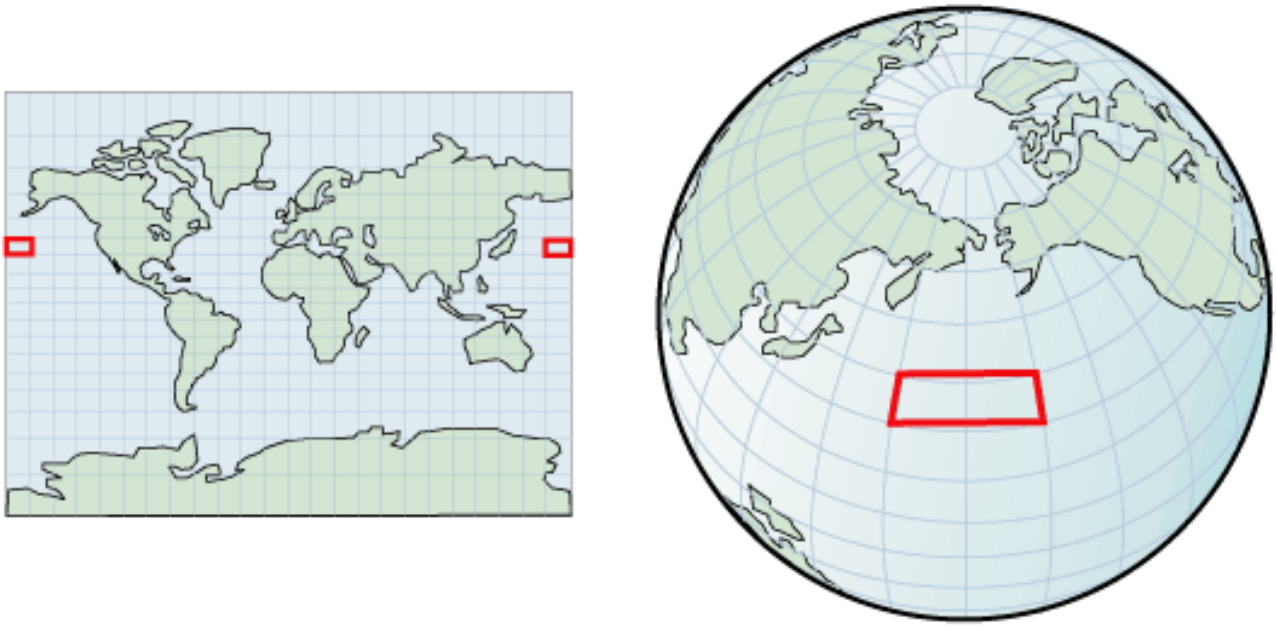


Figura 37. Polígonos que ocupan el meridiano 180º—creación de dos polígonos separados

Si no ha creado dos polígonos separados al utilizar Spatial Extender, se reordenarían los vértices del polígono de modo que se defina un área distinta, tal como muestra la Figura 38 en la página 209. La parte superior de la Figura 38 en la página 209 muestra los vértices correctos de un polígono que ocupa el meridiano 180º:

```
POLYGON((90 0, -90 0, -90 40, 90 40, 90 0))
```

La parte inferior de la Figura 38 en la página 209 muestra los vértices reordenados que producen un polígono que ya no ocupa el meridiano 180º, sino que ahora ocupa el meridiano 0º.

```
POLYGON((-90 0, 90 0, 90 40, -90 40, -90 0))
```

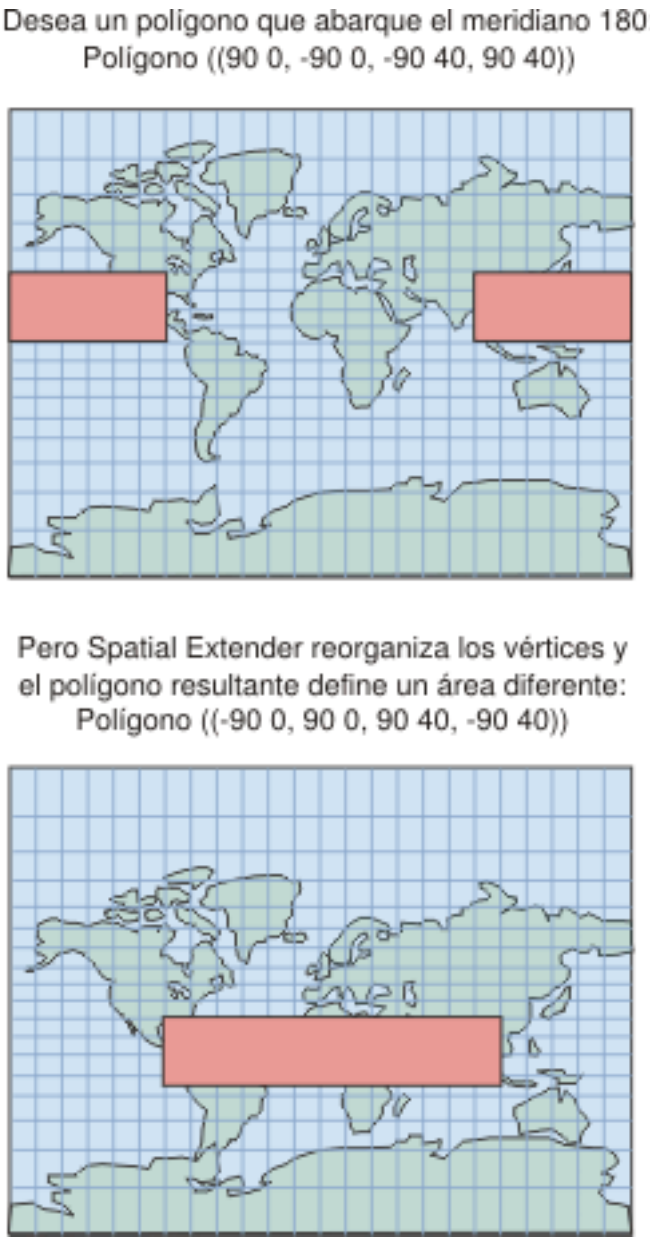


Figura 38. Polígonos que ocupan el meridiano 180º—vértices reordenados

El área definida sería el área complementaria de la tierra, y no el área de interés, tal como se muestra en la Figura 39 en la página 210. De forma similar al ejemplo del segmento lineal anterior, otra forma de manejar esta situación consiste en volver a proyectar los datos de modo que el meridiano 180º no esté en un borde.

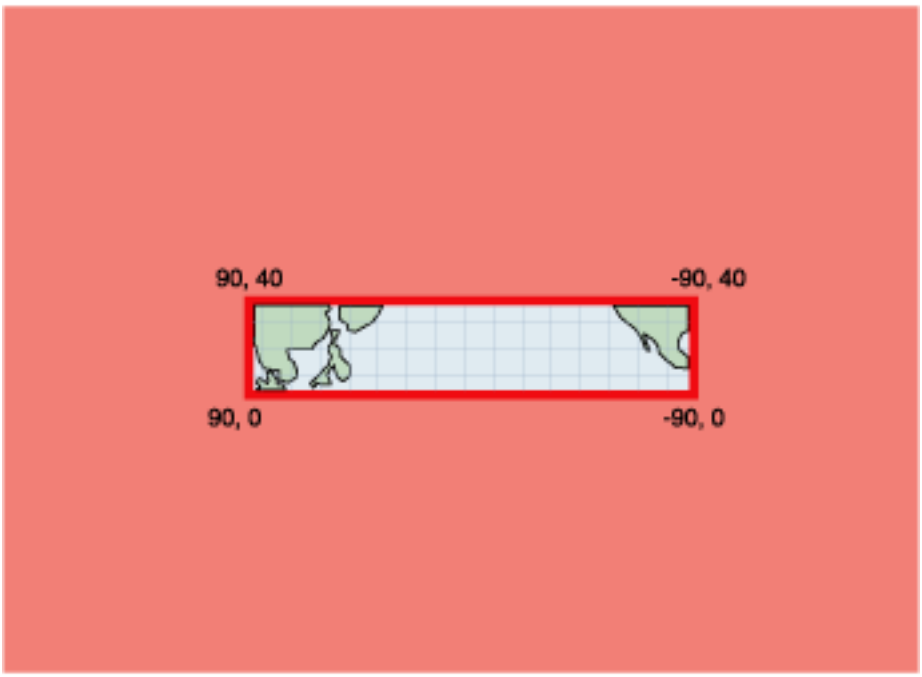


Figura 39. Polígonos que ocupan el meridiano 180º—área complementaria

Polígonos que delimitan un polo

La Figura 40 en la página 211 muestra cómo podría trabajar con un polígono que delimita el polo sur con Spatial Extender o con Geodetic Extender. Como está trabajando directamente en el borde la proyección de tierra plana con Spatial Extender, la distorsión del mapa de la superficie de la tierra le obliga a incorporar bordes y vértices adicionales para representar el polo dentro de un polígono:

```
POLYGON((–180 –90, 180 –90, 180 –60, –180 –60, –180 –90))
```

La representación terrestre redonda (Geodetic Extender) muestra el polígono alrededor del polo sur como un círculo que sigue al paralelo sur –60°:

```
POLYGON((0 –60, –1 –60, –2 –60, ..., –179 –60, 180 –60, 179 –60, ..., 1 –60, 0 –60))
```

Una forma más adecuada de representar este círculo consiste en volver a proyectar los datos de forma que tanto el polo sur como el área circundante sean visibles en el mapa.

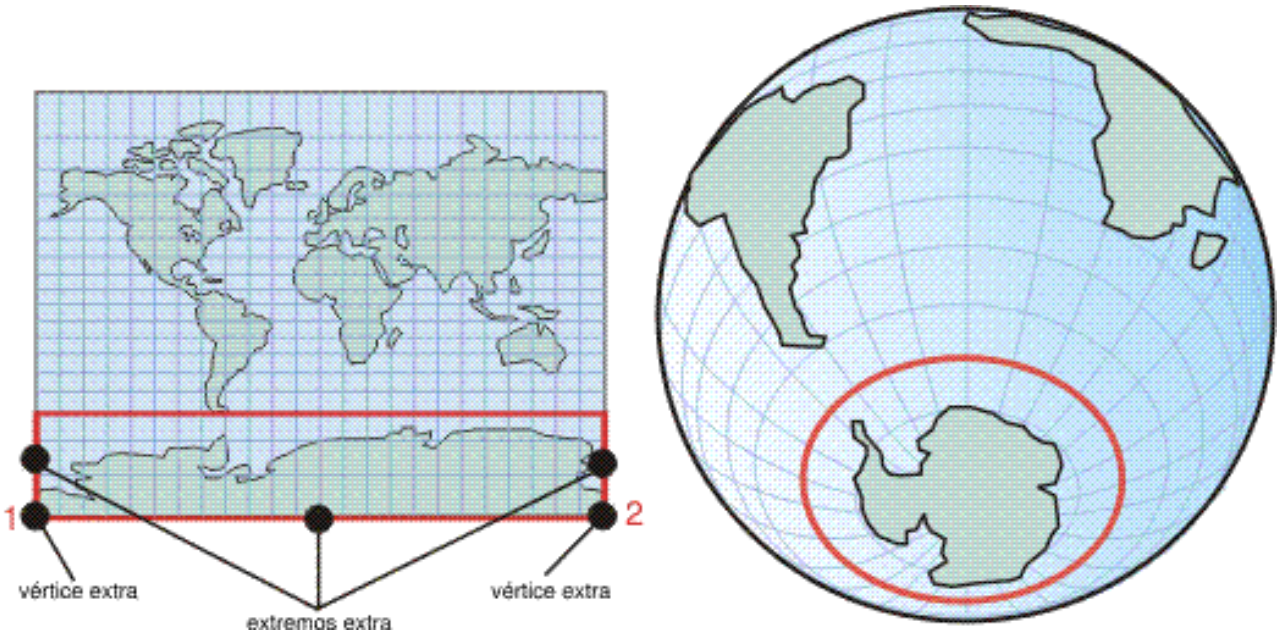


Figura 40. Polígonos que delimitan un polo

En los ejemplos anteriores puede obtener resultados precisos si elige un sistema de referencia espacial proyectado apropiado. Sin embargo, ninguna proyección puede resolverlos todos de forma simultánea. Por ejemplo, una proyección que no tenga el meridiano 180º en el borde coloca a este último en otro lugar y desplaza el área del problema.

Polígonos que representan hemisferios, cinturones ecuatoriales o toda la tierra

Si necesita utilizar un polígono para representar áreas grandes de la superficie de la tierra, como por ejemplo uno de los hemisferios, los cinturones ecuatoriales o la propia tierra al completo, deberá tener en cuenta las distintas formas que tienen Spatial Extender y Geodetic Extender para manejar estos casos. En estas situaciones, una representación terrestre redonda obtiene resultados precisos para cálculos de distancia y área; por el contrario, una cuidadosa elección de la proyección no puede hacerlo.

Por ejemplo, la Figura 41 en la página 212 muestra los polígonos que definen el hemisferio occidental en una representación terrestre plana (Spatial Extender) y en una representación terrestre redonda (Geodetic Extender).

- En la representación terrestre plana de la parte superior de la Figura 41 en la página 212, cuatro coordenadas representan el hemisferio occidental en un formato de texto convencional como 'POLYGON((0 -90, 0 90, -180 90, 180 -90, 0 -90))'.
- En la representación terrestre redonda, cuatro coordenadas representan el hemisferio occidental en un formato de texto convencional como 'POLYGON((0 0, 0 90, 180 0, 0 -90, 0 0))'. Estas cuatro coordenadas definen un anillo alrededor de la tierra a lo largo del meridiano 0º y de su línea de las antípodas, el meridiano 180º.

Si especifica los mismos cuatro puntos en el orden inverso, está definiendo el hemisferio oriental:

- En una representación terrestre plana, el hemisferio oriental es 'POLYGON((0 -90, 180 -90, 180 90, 0 90, 0 -90))'.
- En una representación terrestre redonda, el hemisferio oriental es 'POLYGON((0 -90, 180 0, 0 90, 0 0, 0 -90))'.

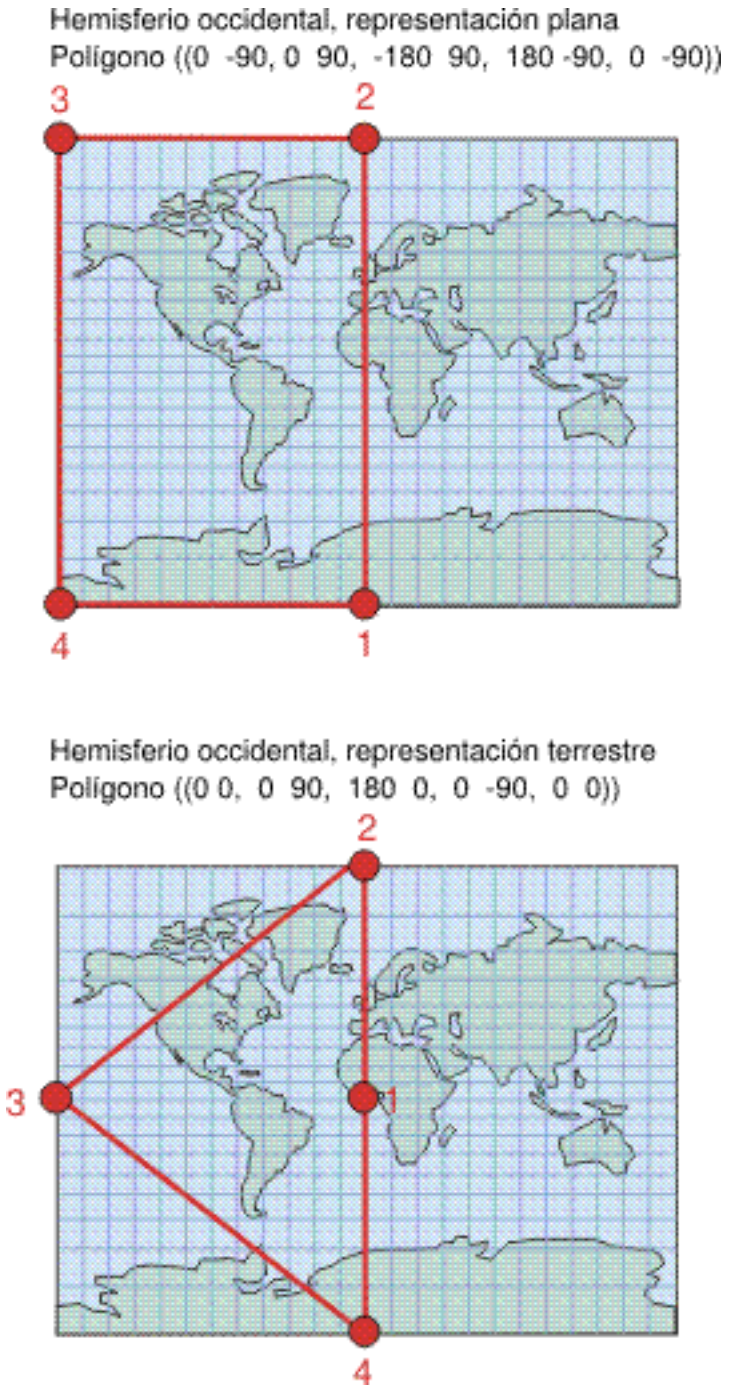


Figura 41. Polígonos que representan hemisferios

La Figura 42 en la página 214 muestra las coordenadas de polígonos que definen el cinturón ecuatorial en una representación terrestre plana (Spatial Extender) y en una representación terrestre redonda (Geodetic Extender).

- La parte superior de la Figura 42 en la página 214 muestra la representación en forma de tierra plana del cinturón ecuatorial con coordenadas en formato de texto convencional como 'POLYGON((180 –60, 180 60, –180 60, –180 –60, 180 –60))'.
- En la representación terrestre redonda de la parte inferior de la Figura 42 en la página 214, se define el área de exclusión de dos anillos para representar el cinturón ecuatorial:

```
'MULTIPOLYGON(((0 60, -120 60, 120 60, 0 60)),
   ((0 -60, 120 -60, -120 -60, 0 -60)))'
```

  Sólo se muestran tres puntos de cada anillo para que quede más claro. En realidad, si desea que los anillos sigan de forma más cercana la línea de latitud 60 o –60, deberá añadir más puntos intermedios. El primer anillo ((0 60, -120 60, 120 60, 0 60)) especifica los vértices en el orden que define el área sur de la línea de latitud 60. El segundo anillo ((0 -60, 120 -60, -120 -60, 0 -60)) especifica el área norte de la línea de latitud –60.

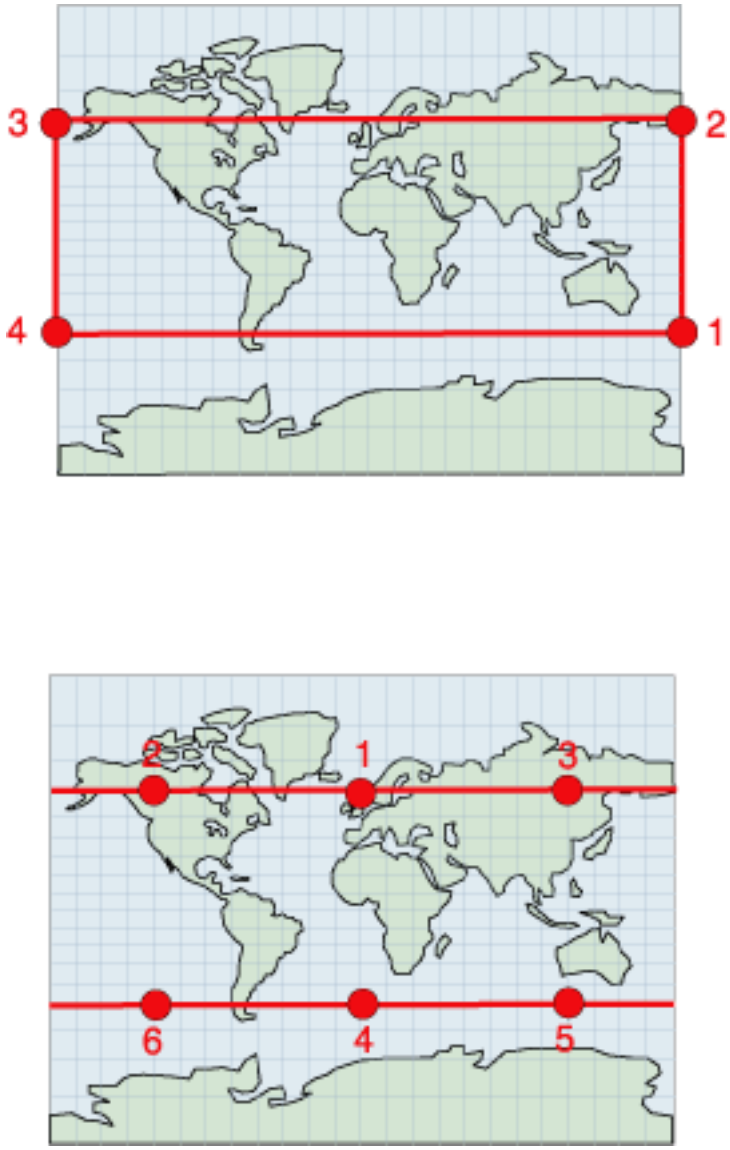


Figura 42. Polígonos que representan cinturones ecuatoriales

La Figura 43 en la página 215 muestra polígonos que definen la tierra al completo en una representación terrestre plana (Spatial Extender) y en una representación terrestre redonda (Geodetic Extender). Ambas representaciones representan toda la tierra con el mismo polígono en formato de texto convencional como 'POLYGON((-180 -90, 180 -90, 180 90, -180 90, -180 -90))'.

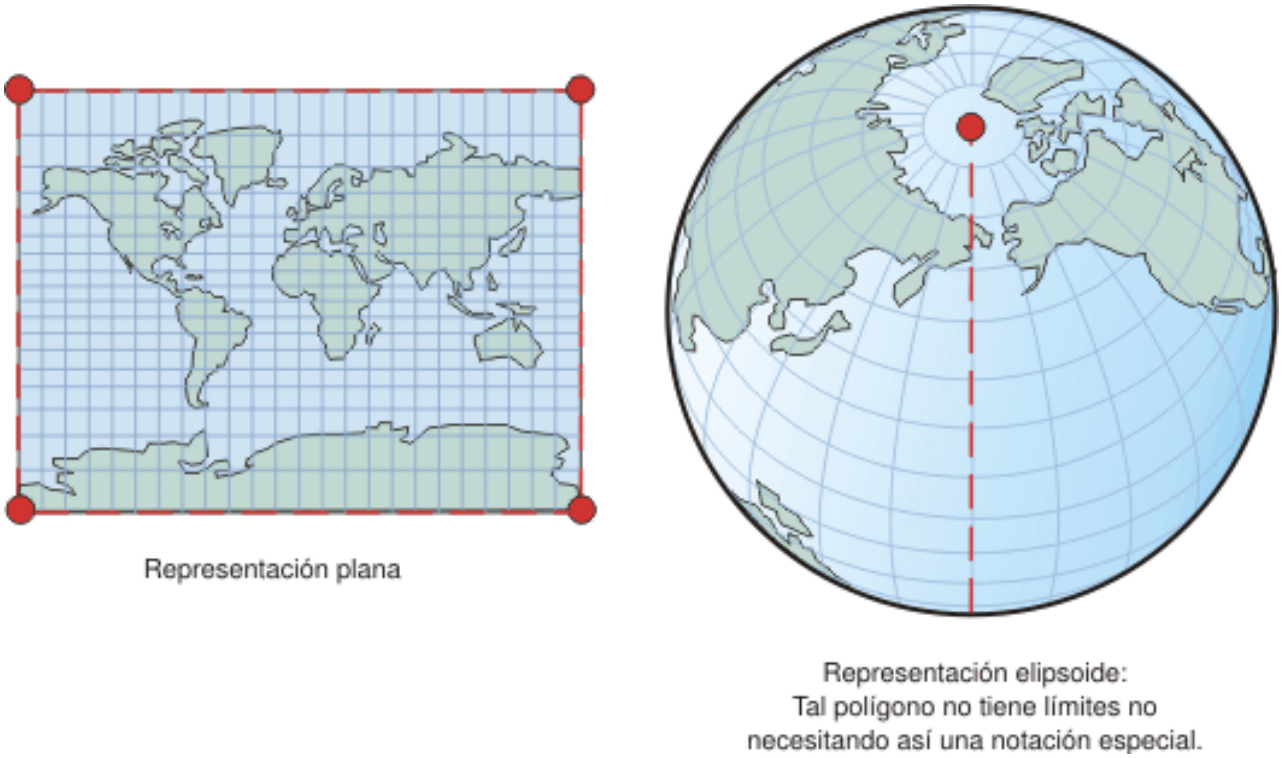


Figura 43. Polígonos que representan toda la tierra

Conceptos relacionados:

- "Cuándo utilizar DB2 Geodetic Extender y DB2 Spatial Extender" en la página 166
- "Regiones geodésicas" en la página 170
- "Latitud y longitud geodésica" en la página 167
- "Distancias geodésicas" en la página 168
- "Esferoides geodésicos" en la página 229

Funciones espaciales soportadas por DB2 Geodetic Extender

DB2 Spatial Extender se basa en la biblioteca de funciones proporcionada por ESRI y DB2 Geodetic Extender se basa en la biblioteca de funciones Hipparchus. Las diferencias de funcionamiento entre las bibliotecas ESRI e Hipparchus se traducen en diferencias menores en cuanto al comportamiento de algunas funciones. La tabla siguiente muestra funciones de Spatial Extender para las que Geodetic Extender tiene soporte e indica las diferencias de comportamiento. Para obtener información sobre el uso y la sintaxis de las funciones espaciales, consulte el apartado correspondiente sobre funciones espaciales.

Tabla 26. Soporte de funciones para Geodetic Extender

Función	¿Soportada por DB2 Geodetic Extender?	Diferencia de comportamiento para DB2 Geodetic Extender
EnvelopesIntersect	Sí	Ninguna
MBR Aggregate (Agregado MBR)	No	No aplicable
ST_AppendPoint	No	No aplicable
ST_Area	Sí	La unidad de medida por omisión es el metro.

Tabla 26. Soporte de funciones para Geodetic Extender (continuación)

Función	¿Soportada por DB2 Geodetic Extender?	Diferencia de comportamiento para DB2 Geodetic Extender
ST_AsBinary	Sí	Ninguno
ST_AsGML	Sí	Ninguno
ST_AsShape	Sí	Ninguno
ST_AsText	Sí	Ninguno
ST_Boundary	No	No aplicable
ST_Buffer	Sí	Soportada únicamente con puntos y multipuntos. La distancia puede ser un valor negativo. La unidad de medida por omisión es el metro.
ST_Centroid	No	No aplicable
ST_ChangePoint	No	No aplicable
ST_Contains	Sí	Ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.
ST_ConvexHull	No	No aplicable
ST_CoordDim	Sí	Ninguno
ST_Crosses	No	No aplicable
ST_Difference	Sí	No soportada con cadenas lineales y multilíneas. Ambas geometrías deben estar en el mismo SRS geodésico. La dimensión de la geometría devuelta es la misma que la de las geometrías de entrada.
ST_Dimension	Sí	Ninguno
ST_Disjoint	Sí	Ninguno
ST_Distance	Sí	Devuelve la *distancia geodésica*. Ambas geometrías deben estar en el mismo SRS geodésico. La unidad de medida por omisión es el metro.
ST_Edge_GC_USA	Sí	Ninguno
ST_Endpoint	Sí	Ninguno
ST_Envelope	Sí	La envoltura es un polígono que delimita el círculo delimitador mínimo (MBC) de la geometría.
ST_EnvIntersects	Sí	Ninguno
ST_EqualCoordsys	Sí	Ninguno
ST_Equals	No	No aplicable
ST_EqualSRS	Sí	Ninguno
ST_ExteriorRing	Sí	Ninguno
ST_FindMeasure o ST_LocateAlong	No	No aplicable
ST_Generalize	Sí	La unidad del umbral es el metro.
ST_GeomCollection	No	No aplicable
ST_GeomCollFromTxt	No	No aplicable
ST_GeomCollFromWKB	No	No aplicable
ST_Geometry	Sí	Ninguno

Tabla 26. Soporte de funciones para Geodetic Extender (continuación)

Función	¿Soportada por DB2 Geodetic Extender?	Diferencia de comportamiento para DB2 Geodetic Extender
ST_GeometryN	Sí	Ninguno
ST_GeometryType	Sí	Ninguno
ST_GeomFromText	Sí	Ninguno
ST_GeomFromWKB	Sí	Ninguno
ST_GetIndexParms	No	No aplicable
ST_InteriorRingN	Sí	Ninguno
ST_Intersection	Sí	La dimensión de la geometría devuelta es la de la entrada con la dimensión inferior, excepto cuando la dimensión de la intersección de dos cadenas sea 0.
ST_Intersects	Sí	Ambas geometrías deben estar en el mismo SRS geodésico.
ST_Is3d	Sí	Ninguno
ST_IsClosed	Sí	Ninguno
ST_IsEmpty	Sí	Ninguno
ST_IsMeasured	Sí	Ninguno
ST_IsRing	No	No aplicable
ST_IsSimple	No	No aplicable
ST_IsValid	Sí	Ninguno
ST_Length	Sí	La unidad de medida por omisión es el metro.
ST_LineFromText	Sí	Ninguno
ST_LineFromWKB	Sí	Ninguno
ST_LineString	Sí	Ninguno
ST_LineStringN	Sí	Ninguno
ST_M	Sí	Ninguna
ST_MaxM	Sí	Ninguno
ST_MaxX	Sí	Devuelve el valor X máximo del círculo delimitador mínimo (MBC). **Nota:** Si el MBC cruza la línea de datos, el valor de ST_MaxX es menor que el valor de ST_MinX. Si el MBC incluye el polo norte, o el polo sur, ST_MinX es –180 y ST_MaxX es 180.
ST_MaxY	Sí	Devuelve el valor Y máximo del MBC. **Nota:** Si el MBC incluye el polo norte, el valor de ST_MaxY es 90.
ST_MaxZ	Sí	Ninguno
ST_MBR	Sí	MBR es una geometría que incluye el MBC de la geometría.
ST_MBRIntersects	Sí	Ninguno
ST_MeasureBetween o ST_LocateBetween	No	No aplicable
ST_MidPoint	Sí	Ninguno
ST_MinM	Sí	Ninguno

Tabla 26. Soporte de funciones para Geodetic Extender (continuación)

Función	¿Soportada por DB2 Geodetic Extender?	Diferencia de comportamiento para DB2 Geodetic Extender
ST_MinX	Sí	Devuelve el valor X mínimo del MBC. **Nota:** Si el MBC cruza la línea de datos, el valor de ST_MinX es mayor que el valor de ST_MaxX. Si el MBC incluye el polo norte, o el polo sur, ST_MinX es –180 y ST_MaxX es 180.
ST_MinY	Sí	Devuelve el valor Y mínimo del MBC. **Nota:** Si el MBC incluye el polo sur, el valor de ST_MinY es –90.
ST_MinZ	Sí	Ninguno
ST_MLineFromText	Sí	Ninguno
ST_MLineFromWKB	Sí	Ninguno
ST_MPointFromText	Sí	Ninguno
ST_MPointFromWKB	Sí	Ninguno
ST_MPolyFromText	Sí	Ninguno
ST_MPolyFromWKB	Sí	Ninguno
ST_MultiLineString	Sí	Ninguno
ST_MultiPoint	Sí	Ninguno
ST_MultiPolygon	Sí	Ninguno
ST_NumGeometries	Sí	Ninguno
ST_NumInteriorRing	Sí	Ninguno
ST_NumLineStrings	Sí	Ninguno
ST_NumPoints	Sí	Ninguno
ST_NumPolygons	Sí	Ninguno
ST_Overlaps	No	No aplicable
ST_Perimeter	Sí	La unidad de medida por omisión es el metro.
ST_PerpPoints	No	No aplicable
ST_Point	Sí	Ninguno
ST_PointFromText	Sí	Ninguno
ST_PointFromWKB	Sí	Ninguno
ST_PointN	Sí	Ninguno
ST_PolyFromText	Sí	Ninguno
ST_PolyFromWKB	Sí	Ninguno
ST_PointOnSurface	Sí	Ninguno
ST_Polygon	Sí	Ninguno
ST_PolygonN	Sí	Ninguno
ST_Relate	No	No aplicable
ST_RemovePoint	No	No aplicable
ST_SrsId or ST_SRID	Sí	Ninguno
ST_SrsName	Sí	Ninguno
ST_StartPoint	Sí	Ninguno

Tabla 26. Soporte de funciones para Geodetic Extender (continuación)

Función	¿Soportada por DB2 Geodetic Extender?	Diferencia de comportamiento para DB2 Geodetic Extender
ST_SymDifference	Sí	No soportada con cadenas lineales y multilíneas. La dimensión de la geometría devuelta es la misma que la de las geometrías de entrada. Ambas geometrías deben estar en el mismo SRS geodésico.
ST_ToGeomColl	No	No aplicable
ST_ToLineString	Sí	Ninguno
ST_ToMultiLine	Sí	Ninguno
ST_ToMultiPoint	Sí	Ninguno
ST_ToPoint	Sí	Ninguno
ST_ToPolygon	Sí	Ninguno
ST_Touches	No	No aplicable
ST_Transform	Sí	Ninguna. **Nota:** Las transformaciones de coordenadas se realizan punto por punto. Al transformar entre sistemas de coordenadas geodésicos y sistemas de coordenadas planas no proyectadas, preste atención a los polígonos y a las cadenas lineales que ocupan el meridiano 180º, o bien delimite uno o ambos polos. Debido a que Spatial Extender y Geodetic Extender tratan estos casos de forma distinta, es posible que las geometrías que son válidas en un sistema de coordenadas de tierra plana no sean válidas en un sistema de tierra redonda y viceversa. Para obtener más información, consulte el apartado "Diferencias de utilización entre las representaciones terrestres planas y redondas" en la página 205.
ST_Union	Sí	Ambas geometrías deben estar en el mismo SRS geodésico.
ST_Within	Sí	Ambas geometrías deben estar en el mismo SRS geodésico.
ST_WKBToSQL	Sí	Ninguno
ST_WKTToSQL	Sí	Ninguno
ST_X	Sí	Ninguno
ST_Y	Sí	Ninguno
ST_Z	Sí	Ninguno
Union Aggregate	No	No aplicable

Conceptos relacionados:

- "Cuándo utilizar DB2 Geodetic Extender y DB2 Spatial Extender" en la página 166
- "Regiones geodésicas" en la página 170
- "Latitud y longitud geodésica" en la página 167
- "Distancias geodésicas" en la página 168

Tareas relacionadas:

- "Creación de índices Voronoi geodésicos" en la página 187

Información relacionada:

- "Diferencias de utilización entre las representaciones terrestres planas y redondas" en la página 205

Procedimientos almacenados y vistas de catálogo de DB2 Geodetic Extender

DB2 Geodetic Extender tiene soporte para las mismas vistas de catálogo que DB2 Spatial Extender y da soporte a un subconjunto de los procedimientos almacenados espaciales.

Geodetic Extender no tiene soporte para los siguientes procedimientos almacenados:

- ST_disable_autogeocoding
- ST_enable_autogeocoding
- ST_register_geocoder
- ST_remove_geocoding_setup
- ST_run_geocoding
- ST_setup_geocoding
- ST_unregister_geocoder

Geodetic Extender proporciona 318 sistemas de referencia espacial geodésicos predefinidos que aparecen en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS. Consulte el apartado "Sistemas de referencia soportados por DB2 Geodetic Extender" para obtener una lista completa.

Sistemas de referencia soportados por DB2 Geodetic Extender

Tal como se describe en el apartado "Sistema de coordenadas geográficas" en la página 60, un *datum* es un conjunto de valores que define la posición de un elipsoide con relación al centro de la tierra. Un sistema de referencia espacial (SRS) es un conjunto de parámetros que asocian un datum con un elipsoide y se identifica con un identificador de sistema de referencia espacial (SRID). La Tabla 28 enumera los sistemas de referencia predefinidos que proporciona DB2 Geodetic Extender. Los valores de desplazamiento y los factores de escala de todos los SRS geodésicos predefinidos son los mismos. La tabla siguiente muestra sus valores.

Tabla 27. Valores de desplazamiento y de escala de los SRS geodésicos predefinidos

Parámetro de SRS	Valor
xOffset	–180
yOffset	–90
zOffset	–50000
mOffset	–1000
xScale	5965232
yScale	5965232
zScale	1000
mScale	1000

El parámetro yScale es siempre el mismo que xScale.

Puede elegir cualquier datum que aparezca en la Tabla 28 para el sistema de referencia espacial. Elija el que se mejor se ajuste a sus datos. Por ejemplo, uno de los puntos de referencia más utilizados habitualmente, World Geodetic System 1984 (WGS 1984), toma el centro de la tierra como punto de origen y genera un mapa de todo el globo; se trata de un datum centrado en la tierra. A diferencia de este, un datum regional como North American 1927 genera un mapa de Norteamérica empezando desde un punto del suelo. Un datum regional resulta preciso para la región que pretende modelar, pero se necesita un datum geodésico centrado en la tierra para manejar ubicaciones en todo el globo.

Tabla 28. SRID con datum asociado y elipsoide

SRID	Nombre del datum	Elipsoide de referencia
2000000000	WGS 1984	WGS 1984
2000000001	Abidjan 1987	Clarke 1880 (RGS)
2000000002	Accra	War Office
2000000003	Adindan	Clarke 1880 (RGS)
2000000004	Afgooye	Krasovsky 1940
2000000005	Agadez	Clarke 1880 (IGN)
2000000006	Australian Geodetic Datum 1966	Australian
2000000007	Australian Geodetic Datum 1984	Australian
2000000008	Ain el Abd 1970	International 1924
2000000009	Airy 1830	Airy 1830
2000000010	Airy Modified	Airy Modified
2000000011	Islas de Alaska	Clarke 1866
2000000012	Amersfoort	Bessel 1841
2000000013	Anguilla 1957	Clarke 1880 (RGS)
2000000014	Anna 1 Astro 1965	Australian
2000000015	Antigua Astro 1943	Clarke 1880 (RGS)
2000000016	Aratu	International 1924
2000000017	Arc 1950	Clarke 1880 (Arc)
2000000018	Arc 1960	Clarke 1880 (RGS)
2000000019	Ascension Island 1958	International 1924
2000000020	Assumed Geographic (NAD27 para archivos de formas sin PRJ)	Clarke 1866
2000000021	Astronomical Station 1952	International 1924
2000000022	ATF (Paris)	Plessis 1817
2000000023	Average Terrestrial System 1977	ATS 1977
2000000024	Australian National	Australian
2000000025	Ayabelle Lighthouse	Clarke 1880 (RGS)
2000000026	Bab South Astro (Bablethuap Is, República de Palau)	Clarke 1866
2000000027	Barbados 1938	Clarke 1880 (RGS)
2000000028	Batavia	Bessel 1841

Tabla 28. SRID con datum asociado y elipsoide (continuación)

SRID	Nombre del datum	Elipsoide de referencia
2000000029	Batavia (Yakarta)	Bessel 1841
2000000030	Astro Beacon E 1945	International 1924
2000000031	Beduaram	Clarke 1880 (IGN)
2000000032	Beijing 1954	Krasovsky 1940
2000000033	Reseau National Belge 1950	International 1924
2000000034	Belge 1950 (Bruselas)	International 1924
2000000035	Reseau National Belge 1972	International 1924
2000000036	Bellevue (IGN)	International 1924
2000000037	Bermuda 1957	Clarke 1866
2000000038	Bern 1898	Bessel 1841
2000000039	Bern 1898 (Berna)	Bessel 1841
2000000040	Bern 1938	Bessel 1841
2000000041	Bessel 1841	Bessel 1841
2000000042	Bessel Modified	Bessel Modified
2000000043	Bessel Namibia	Bessel Namibia
2000000044	Bissau	International 1924
2000000045	Bogota	International 1924
2000000046	Bogota (Bogotá)	International 1924
2000000047	Bukit Rimpah	Bessel 1841
2000000048	Camacupa	Clarke 1880 (RGS)
2000000049	Campo Inchauspe	International 1924
2000000050	Camp Area Astro	International 1924
2000000051	Canton Astro 1966	International 1924
2000000052	Cape	Clarke 1880 (Arc)
2000000053	Cape Canaveral	Clarke 1866
2000000054	Carthage	Clarke 1880 (IGN)
2000000055	Carthage (grados)	Clarke 1880 (IGN)
2000000056	Carthage (París)	Clarke 1880 (IGN)
2000000057	CH 1903	Bessel 1841
2000000058	CH 1903+	Bessel 1841
2000000059	Chatham Island Astro 1971	International 1924
2000000060	Chos Malal 1914	International 1924
2000000061	Swiss Terrestrial Ref. Frame 1995	GRS 1980
2000000062	Chua	International 1924
2000000063	Clarke 1858	Clarke 1858
2000000064	Clarke 1866	Clarke 1866
2000000065	Clarke 1866 (Michigan)	Clarke 1866 (Michigan)
2000000066	Clarke 1880	Clarke 1880

Tabla 28. SRID con datum asociado y elipsoide (continuación)

SRID	Nombre del datum	Elipsoide de referencia
2000000067	Clarke 1880 (Arc)	Clarke 1880 (Arc)
2000000068	Clarke 1880 (Benoit)	Clarke 1880 (Benoit)
2000000069	Clarke 1880 (IGN)	Clarke 1880 (IGN)
2000000070	Clarke 1880 (RGS)	Clarke 1880 (RGS)
2000000071	Clarke 1880 (SGA)	Clarke 1880 (SGA)
2000000072	Conakry 1905	Clarke 1880 (IGN)
2000000073	Corrego Alegre	International 1924
2000000074	Cote d'Ivoire	Clarke 1880 (IGN)
2000000075	Dabola 1981	Clarke 1880 (RGS)
2000000076	Datum 73	International 1924
2000000077	Dealul Piscului 1933 (Rumanía)	International 1924
2000000078	Dealul Piscului 1970 (Rumanía)	Krasovsky 1940
2000000079	Deception Island	Clarke 1880 (RGS)
2000000080	Deir ez Zor	Clarke 1880 (IGN)
2000000081	Deutsche Hauptdreiecksnetz	Bessel 1841
2000000082	Dominica 1945	Clarke 1880 (RGS)
2000000083	DOS 1968	International 1924
2000000084	Astro DOS 71/4	International 1924
2000000085	Douala	Clarke 1880 (IGN)
2000000086	Easter Island 1967	International 1924
2000000087	European Datum 1950	International 1924
2000000088	European Datum 1950 (ED77)	International 1924
2000000089	European Datum 1987	International 1924
2000000090	Egypt 1907	Helmert 1906
2000000091	Estonia 1937	Bessel 1841
2000000092	Estonia 1992	GRS 1980
2000000093	European Terrestrial Ref. Frame 1989	WGS 1984
2000000094	European 1979	International 1924
2000000095	European Libyan Datum 1979	International 1924
2000000096	Everest 1830	Everest 1830
2000000097	Everest (Bangladesh)	Everest Adjustment 1937
2000000098	Everest (Definition 1962)	Everest (Definition 1962)
2000000099	Everest (Definition 1967)	Everest (Definition 1967)
2000000100	Everest (Definition 1975)	Everest (Definition 1975)
2000000101	Everest (India y Nepal)	Everest (Definition 1962)
2000000102	Everest 1830 Modified	Everest 1830 Modified

Tabla 28. SRID con datum asociado y elipsoide (continuación)

SRID	Nombre del datum	Elipsoide de referencia
2000000103	Everest Modified 1969	Everest Modified 1969
2000000104	Fahud	Clarke 1880 (RGS)
2000000105	Final Datum 1958	Clarke 1880 (RGS)
2000000106	Fischer 1960	Fischer 1960
2000000107	Fischer 1968	Fischer 1968
2000000108	Fischer Modified	Fischer Modified
2000000109	Fort Thomas 1955	Clarke 1880 (RGS)
2000000110	Gandajika 1970	International 1924
2000000111	Gan 1970	International 1924
2000000112	Garoua	Clarke 1880 (IGN)
2000000113	Geocentric Datum of Australia 1994	GRS 1980
2000000114	GEM 10C Gravity Potential Model	GEM 10C
2000000115	Greek Geodetic Ref. System 1987	GRS 1980
2000000116	Graciosa Base SW 1948	International 1924
2000000117	Greek	Bessel 1841
2000000118	Greek (Atenas)	Bessel 1841
2000000119	Grenada 1953	Clarke 1880 (RGS)
2000000120	GRS 1967	GRS 1967
2000000121	GRS 1980	GRS 1980
2000000122	Guam 1963	Clarke 1866
2000000123	Gunung Segara	Bessel 1841
2000000124	GUX 1 Astro	International 1924
2000000125	Guyane Francaise	International 1924
2000000126	Hanoi 1972	Krasovsky 1940
2000000127	Hartebeesthoek 1994	WGS 1984
2000000128	Helmert 1906	Helmert 1906
2000000129	Herat North	International 1924
2000000130	Hermannskogel	Bessel 1841
2000000131	Hito XVIII 1963	International 1924
2000000132	Hjorsey 1955	International 1924
2000000133	Hong Kong 1963	International 1924
2000000134	Hong Kong 1980	International 1924
2000000135	Hough 1960	Hough 1960
2000000136	Hungarian Datum 1972	GRS 1967
2000000137	Hu Tzu Shan	International 1924
2000000138	Indian 1954	Everest Adjustment 1937
2000000139	Indian 1960	Everest Adjustment 1937

Tabla 28. SRID con datum asociado y elipsoide (continuación)

SRID	Nombre del datum	Elipsoide de referencia
2000000140	Indian 1975	Everest Adjustment 1937
2000000141	Indonesian National	Indonesian National
2000000142	Indonesian Datum 1974	Indonesian
2000000143	International 1927	International 1924
2000000144	International 1967	International 1967
2000000145	IRENET95	GRS 1980
2000000146	Israel	GRS 1980
2000000147	ISTS 061 Astro 1968	International 1924
2000000148	ISTS 073 Astro 1969	International 1924
2000000149	Jamaica 1875	Clarke 1880
2000000150	Jamaica 1969	Clarke 1866
2000000151	Japan Geodetic Datum 2000	GRS 1980
2000000152	Johnston Island 1961	International 1924
2000000153	Kalianpur 1880	Everest 1830
2000000154	Kalianpur 1937	Everest Adjustment 1937
2000000155	Kalianpur 1962	Everest (Definition 1962)
2000000156	Kalianpur 1975	Everest (Definition 1975)
2000000157	Kandawala	Everest Adjustment 1937
2000000158	Kerguelen Island 1949	International 1924
2000000159	Kertau	Everest 1830 Modified
2000000160	Kartastokoordinaattijarjestelma	International 1924
2000000161	Kuwait Oil Company	Clarke 1880 (RGS)
2000000162	Korean Datum 1985	Bessel 1841
2000000163	Korean Datum 1995	WGS 1984
2000000164	Krasovsky 1940	Krasovsky 1940
2000000165	Kuwait Utility	GRS 1980
2000000166	Kusaie Astro 1951	International 1924
2000000167	Lake	International 1924
2000000168	La Canoa	International 1924
2000000169	L.C. 5 Astro 1961	Clarke 1866
2000000170	Leigon	Clarke 1880 (RGS)
2000000171	Liberia 1964	Clarke 1880 (RGS)
2000000172	Datum Lisboa Bessel	Bessel 1841
2000000173	Datum Lisboa Hayford	International 1924
2000000174	Lisbon	International 1924
2000000175	Lisbon (Lisboa)	International 1924

Tabla 28. SRID con datum asociado y elipsoide (continuación)

SRID	Nombre del datum	Elipsoide de referencia
2000000176	LKS 1994	GRS 1980
2000000177	Locodjo 1965	Clarke 1880 (RGS)
2000000178	Loma Quintana	International 1924
2000000179	Lome	Clarke 1880 (IGN)
2000000180	Luzon 1911	Clarke 1866
2000000181	Madrid 1870 (Merid. principal de Madrid)	Struve 1860
2000000182	Madzansua	Clarke 1866
2000000183	Mahe 1971	Clarke 1880 (RGS)
2000000184	Majuro (República de Marshall Is.)	Clarke 1866
2000000185	Makassar	Bessel 1841
2000000186	Makassar (Yakarta)	Bessel 1841
2000000187	Malongo 1987	International 1924
2000000188	Manoca	Clarke 1880 (RGS)
2000000189	Massawa	Bessel 1841
2000000190	Merchich	Clarke 1880 (IGN)
2000000191	Merchich (grados)	Clarke 1880 (IGN)
2000000192	Militar-Geographische Institut	Bessel 1841
2000000193	MGI (Ferro)	Bessel 1841
2000000194	Mhast	International 1924
2000000195	Midway Astro 1961	International 1924
2000000196	Minna	Clarke 1880 (RGS)
2000000197	Monte Mario	International 1924
2000000198	Monte Mario (Rome)	International 1924
2000000199	Montserrat Astro 1958	Clarke 1880 (RGS)
2000000200	Mount Dillon	Clarke 1858
2000000201	Moznet	WGS 1984
2000000202	M'poraloko	Clarke 1880 (IGN)
2000000203	North American Datum 1927	Clarke 1866
2000000204	NAD 1927 CGQ77	Clarke 1866
2000000205	NAD 1927 (1976)	Clarke 1866
2000000206	North American Datum 1983	GRS 1980
2000000207	NAD 1983 (Sistema de ref. espaciales canadiense)	GRS 1980
2000000208	North American Datum 1983 (HARN)	GRS 1980
2000000209	NAD Michigan	Clarke 1866 (Michigan)
2000000210	Nahrwan 1967	Clarke 1880 (RGS)
2000000211	Naparima 1955	International 1924
2000000212	Naparima 1972	International 1924
2000000213	Nord de Guerre (París)	Plessis 1817

Tabla 28. SRID con datum asociado y elipsoide (continuación)

SRID	Nombre del datum	Elipsoide de referencia
2000000214	National Geodetic Network (Kuwait)	WGS 1984
2000000215	NGO 1948	Bessel Modified
2000000216	NGO 1948 (Oslo)	Bessel Modified
2000000217	Nord Sahara 1959	Clarke 1880 (RGS)
2000000218	NSWC 9Z-2	NWL 9D
2000000219	Nouvelle Triangulation Francaise (grados)	Clarke 1880 (IGN)
2000000220	NTF (París) (grads)	Clarke 1880 (IGN)
2000000221	NWL 9D Transit Precise Ephemeris	NWL 9D
2000000222	New Zealand Geodetic Datum 1949	International 1924
2000000223	New Zealand Geodetic Datum 2000	GRS 1980
2000000224	Observatario	Clarke 1866
2000000225	Observ. Meteorologico 1939	International 1924
2000000226	Old Hawaiian	Clarke 1866
2000000227	Oman	Clarke 1880 (RGS)
2000000228	OSGB 1936	Airy 1830
2000000229	OSGB 1970 (SN)	Airy 1830
2000000230	OSU 1986 Geoidal Model	OSU 86F
2000000231	OSU 1991 Geoidal Model	OSU 91A
2000000232	OS (SN) 1980	Airy 1830
2000000233	Padang 1884	Bessel 1841
2000000234	Padang 1884 (Yakarta)	Bessel 1841
2000000235	Palestine 1923	Clarke 1880 (Benoit)
2000000236	Pampa del Castillo	International 1924
2000000237	PDO Survey Datum 1993	Clarke 1880 (RGS)
2000000238	Pico de Las Nieves	International 1924
2000000239	Pitcairn Astro 1967	International 1924
2000000240	Plessis 1817	Plessis 1817
2000000241	Pohnpei (Estados Fed. de Micronesia)	Clarke 1866
2000000242	Point 58	Clarke 1880 (RGS)
2000000243	Pointe Noire	Clarke 1880 (IGN)
2000000244	Porto Santo 1936	International 1924
2000000245	POSGAR	GRS 1980
2000000246	Provisional South Amer. Datum 1956	International 1924
2000000247	Puerto Rico	Clarke 1866
2000000248	Pulkovo 1942	Krasovsky 1940
2000000249	Pulkovo 1995	Krasovsky 1940
2000000250	Qatar 1974	International 1924
2000000251	Qatar 1948	Helmert 1906
2000000252	Qornoq	International 1924

Tabla 28. SRID con datum asociado y elipsoide (continuación)

SRID	Nombre del datum	Elipsoide de referencia
2000000253	Rassadiran	International 1924
2000000254	REGVEN	GRS 1980
2000000255	Reunion	International 1924
2000000256	Reseau Geodesique Francais 1993	GRS 1980
2000000257	RT38	Bessel 1841
2000000258	RT38 (Estocolmo)	Bessel 1841
2000000259	RT 1990	Bessel 1841
2000000260	S-42 Hungary	Krasovsky 1940
2000000261	South American Datum 1969	GRS 1967 Truncated
2000000262	Samboja	Bessel 1841
2000000263	American Samoa 1962	Clarke 1866
2000000264	Santo DOS 1965	International 1924
2000000265	Sao Braz	International 1924
2000000266	Sapper Hill 1943	International 1924
2000000267	Schwarzeck	Bessel Namibia
2000000268	Segora	Bessel 1841
2000000269	Selvagem Grande 1938	International 1924
2000000270	Serindung	Bessel 1841
2000000271	Sierra Leone 1924	War Office
2000000272	Sierra Leone 1960	Clarke 1880 (RGS)
2000000273	Sierra Leone 1968	Clarke 1880 (RGS)
2000000274	SIRGAS	GRS 1980
2000000275	South Yemen	Krasovsky 1940
2000000276	Authalic sphere	Sphere
2000000277	Authalic sphere (ARC/INFO)	Sphere ARC INFO
2000000278	Struve 1860	Struve 1860
2000000279	St. George Island (Alaska)	Clarke 1866
2000000280	St. Kitts 1955	Clarke 1880 (RGS)
2000000281	St. Lawrence Island (Alaska)	Clarke 1866
2000000282	St. Lucia 1955	Clarke 1880 (RGS)
2000000283	St. Paul Island (Alaska)	Clarke 1866
2000000284	St. Vincent 1945	Clarke 1880 (RGS)
2000000285	Sudan	Clarke 1880 (IGN)
2000000286	South Asia Singapore	Fischer Modified
2000000287	S-JTSK	Bessel 1841
2000000288	S-JTSK (Ferro)	Bessel 1841
2000000289	Tananarive 1925	International 1924
2000000290	Tananarive 1925 (Paris)	International 1924
2000000291	Tern Island Astro 1961	International 1924

Tabla 28. SRID con datum asociado y elipsoide (continuación)

SRID	Nombre del datum	Elipsoide de referencia
2000000292	Tete	Clarke 1866
2000000293	Timbalai 1948	Everest (Definition 1967)
2000000294	TM65	Airy Modified
2000000295	TM75	Airy Modified
2000000296	Tokyo	Bessel 1841
2000000297	Trinidad 1903	Clarke 1858
2000000298	Tristan Astro 1968	International 1924
2000000299	Trucial Coast 1948	Helmert 1906
2000000300	Viti Levu 1916	Clarke 1880 (RGS)
2000000301	Voirol 1875	Clarke 1880 (IGN)
2000000302	Voirol 1875 (grados)	Clarke 1880 (IGN)
2000000303	Voirol 1875 (París)	Clarke 1880 (IGN)
2000000304	Voirol Unifie 1960	Clarke 1880 (RGS)
2000000305	Voirol Unifie 1960 (grados)	Clarke 1880 (RGS)
2000000306	Voirol Unifie 1960 (París)	Clarke 1880 (RGS)
2000000307	Wake-Eniwetok 1960	Hough 1960
2000000308	Wake Island Astro 1952	International 1924
2000000309	Walbeck	Walbeck
2000000310	War Office	War Office
2000000311	WGS 1966	WGS 1966
2000000312	WGS 1972	WGS 1972
2000000313	WGS 1972 Transit Broadcast Ephemeris	WGS 1972
2000000314	Yacare	International 1924
2000000315	Yemen Nat'l Geodetic Network 1996	WGS 1984
2000000316	Yoff	Clarke 1880 (IGN)
2000000317	Zanderij	International 1924

Esferoides geodésicos

Un esferoide (también denominado elipsoide) es la parte de un sistema de coordenadas geográficas que define la forma de la superficie de la tierra en una ubicación determinada.

La definición de un sistema de coordenadas incluye la definición de un elipsoide en la definición de SPHEROID que forma parte de la definición de DATUM, tal como muestra el ejemplo siguiente:

```
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199432955]]
```

Para obtener una lista de los esferoide que proporcionan Spatial Extender y Geodetic Extender, consulte el apartado "Sistemas de coordenadas soportados" en la página 539

la página 539. Puede utilizar la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS para recuperar esta información. La columna **DEFINITION** de la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS contiene los valores de las columnas **Nombre**, **Eje semimayor** y **Aplanamiento** de la tabla Esferoides soportados.

Conceptos relacionados:

- "Sistema de coordenadas geográficas" en la página 60

Información relacionada:

- "Vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS" en la página 295
- "ST_create_coordsys" en la página 244

Parte 5. Información de consulta

Capítulo 20. Procedimientos almacenados

Esta sección proporciona información de consulta acerca de los procedimientos almacenados que puede utilizar para configurar DB2 Spatial Extender y crear proyectos que utilicen datos espaciales. Cuando configure DB2 Spatial Extender o cree proyectos desde el Centro de Control de DB2 o desde el procesador de línea de mandatos de DB2, puede invocar estos procedimientos almacenados implícitamente. Por ejemplo, cuando pulsa **Bien** en la ventana DB2 Spatial Extender en el Centro de Control de DB2, DB2 llama al procedimiento almacenado que está asociado con esta ventana.

De forma alternativa, puede invocar el procedimiento almacenado de forma explícita en un programa de aplicación.

Antes de invocar la mayoría de procedimientos almacenados de DB2 Spatial Extender en la base de datos, debe habilitar esta base de datos para operaciones espaciales invocando el procedimiento almacenado ST_enable_db, bien directamente o utilizando el Centro de Control de DB2. (Puede leer acerca de cómo invocar este procedimiento almacenado en el apartado acerca de ST_enable_db, más adelante en esta sección.)

Después de habilitar una base de datos para operaciones espaciales, puede invocar cualquiera de los procedimientos almacenados de DB2 Spatial Extender, bien implícita o explícitamente, en esta base de datos si está conectado a esta base de datos.

Este capítulo proporciona temas para todos los procedimientos almacenados de DB2 Spatial Extender, de la forma siguiente:

- GSE_export_sde
- GSE_import_sde
- ST_alter_coordsys
- ST_alter_srs
- ST_create_coordsys
- ST_create_srs
- ST_disable_autogeocoding
- ST_disable_db
- ST_drop_coordsys
- ST_drop_srs
- ST_enable_autogeocoding
- ST_enable_db
- ST_export_shape
- ST_import_shape
- ST_register_geocoder
- ST_register_spatial_column
- ST_remove_geocoding_setup
- ST_run_geocoding
- ST_setup_geocoding
- ST_unregister_geocoder

- ST_unregister_spatial_column

Las implementaciones de los procedimientos almacenados se archivan en la biblioteca db2gse en el servidor de DB2 Spatial Extender.

GSE_export_sde

Utilice este procedimiento almacenado para exportar una columna espacial y su tabla asociada a un archivo de transferencia SDE.

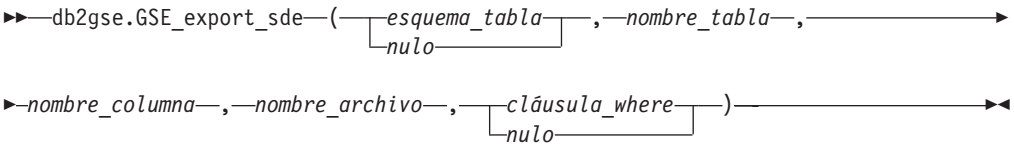
Restricciones:

- Debe existir exactamente una columna espacial en la tabla o vista.
- La columna espacial debe estar registrada.
- No puede realizar adiciones a archivos SDE existentes.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización SYSADM o DBADM. Además, el ID de usuario bajo el que se realiza la acción debe tener el privilegio SELECT sobre la tabla que se va a exportar.

Sintaxis:



Descripciones de parámetros:

esquema_tabla
Especifica el esquema al que pertenece la tabla que está especificada en el parámetro *nombre_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_tabla
Especifica el nombre no calificado de la tabla que está exportando. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_columna
Especifica la columna espacial registrada que está exportando. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_archivo
Especifica el archivo de transferencia SDE al que se va a exportar la columna espacial especifica y su tabla asociada. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es VARCHAR(256).

cláusula_where
Especifica el cuerpo de la cláusula WHERE de SQL, que define una restricción sobre el conjunto de registros que se van a exportar. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se define ninguna restricción en la cláusula WHERE.

Si este parámetro está especificado, el valor puede hacer referencia a cualquier atributo de la tabla que esté exportando.

El tipo de datos de este parámetro es VARCHAR(1024).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado GSE_export_sde. Este ejemplo utiliza un mandato CALL de DB2 para exportar datos desde una tabla denominada CUSTOMERS a archivos SDE:

```
call db2gse.GSE_export_sde(NULL,'CUSTOMERS','LOCATION','/tmp/export_sde_file',
        NULL,?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "GSE_import_sde" en la página 236

GSE_import_sde

Utilice este procedimiento almacenado para importar un archivo de transferencia SDE a una base de datos que esté habilitada para operaciones espaciales. El procedimiento almacenado puede funcionar de cualquiera de estas dos maneras:

- Si el archivo de transferencia SDE está destinado para una tabla existente que tiene una columna espacial registrada, DB2 Spatial Extender carga la tabla con los datos del archivo.
- En caso contrario, DB2 Spatial Extender crea una tabla que tiene una columna espacial, registra esta columna y carga la columna espacial y las otras columnas de la tabla con los datos del archivo.

El sistema de referencia espacial que se especifica en el archivo de transferencia SDE se compara con los sistemas de referencia espacial que están registrados en DB2 Spatial Extender. Si el sistema especificado coincide con un sistema registrado, todos los valores de datos en los datos de transferencia, cuando se carguen, se modificarán de la manera que especifique el sistema registrado. Si el sistema especificado no coincide con ninguno de los sistemas registrados, DB2 Spatial Extender crea un nuevo sistema de referencia espacial para especificar las modificaciones.

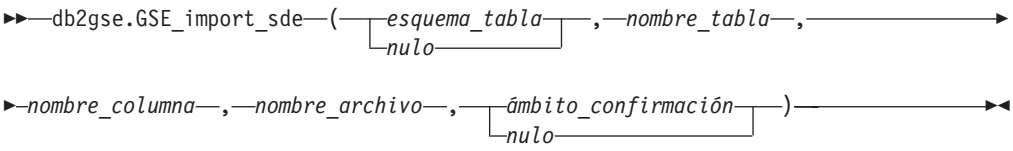
Autorización:

Cuando importa datos a una tabla existente, el ID de usuario utilizado para invocar a 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 a la que se van a importar los datos
- Privilegio CONTROL sobre esta tabla

Cuando se debe crear la tabla a la que desea importar los datos, el ID de usuario utilizado para invocar a este procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos que contiene la tabla que se va a crear.

Sintaxis:



Descripciones de parámetros:

esquema_tabla
Especifica el esquema al que pertenece la tabla o vista que está especificada en el parámetro *nombre_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_tabla
Especifica el nombre no calificado de la tabla en la que se cargarán los datos de transferencia SDE. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_columna
Especifica la columna registrada en la que se cargarán los datos espaciales del archivo de transferencia SDE. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_archivo
Especifica el archivo de transferencia SDE que se va a importar. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es VARCHAR(256).

ámbito_confirmación
Especifica el número de registros que se importarán antes de emitir un COMMIT. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utiliza un valor de 0 (cero) y no se confirma ningún registro.

El tipo de datos de este parámetro es INTEGER.

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado GSE_import_sde. Este ejemplo utiliza

un mandato CALL de DB2 para importar un archivo SDE denominado tmp/clienteSDE a la tabla denominada CUSTOMERS. Este mandato CALL especifica que se va a realizar un COMMIT después de cada 5 registros geocodificados:

```
call db2gse.GSE_import_sde(NULL,'CUSTOMERS','LOCATION',
     '/tmp/clienteSde', 5, ?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "GSE_export_sde" en la página 234

ST_alter_coordsys

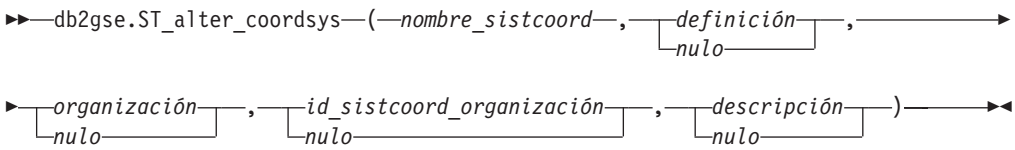
Utilice este procedimiento almacenado para actualizar una definición de sistema de coordenadas en la base de datos. Cuando se procesa este sistema almacenado, la información sobre el sistema de coordenadas se actualiza en la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Atención: Tenga cuidado al utilizar este procedimiento almacenado. Si utiliza este procedimiento almacenado para cambiar la definición del sistema de coordenadas y tiene datos espaciales existentes que están asociados con un sistema de referencia espacial que está basado en este sistema de coordenadas, es posible que sin darse cuenta modifique los datos espaciales. Si los datos espaciales resultan afectados, el usuario es responsable de asegurar que los datos espaciales modificados sigan siendo precisos y válidos.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización SYSADM o DBADM.

Sintaxis:



Descripciones de parámetros:

nombre_sistcoord
Identifica de forma exclusiva el sistema de coordenadas. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

definición
Define el sistema de coordenadas. Aunque debe especificar un valor para este

parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, la definición del sistema de coordenadas no cambia.

El tipo de datos de este parámetro es VARCHAR(2048).

organización
Especifica la organización que ha definido el sistema de coordenadas y proporcionado la definición para la misma; por ejemplo, "European Petroleum Survey Group (EPSG)". Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si este parámetro tiene un valor nulo, la organización del sistema de coordenadas no cambia. Si este parámetro no tiene un valor nulo, el parámetro *id_sistcoord_organización* no puede tener un valor nulo; en este caso, la combinación de los parámetros *organización* e *id_sistcoord_organización* identifica exclusivamente el sistema de coordenadas.

El tipo de datos de este parámetro es VARCHAR(128).

id_sistcoord_organización
Especifica un identificador numérico que la entidad que se lista en el parámetro *organización* asigna a este sistema de coordenadas. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si este parámetro tiene un valor nulo, el parámetro *organización* debe tener también un valor nulo; en este caso, el identificador del sistema de coordenadas de la organización no cambia. Si este parámetro no tiene un valor nulo, el parámetro *organización* no puede tener un valor nulo; en este caso, la combinación de los parámetros *organización* e *id_sistcoord_organización* identifica exclusivamente el sistema de coordenadas.

El tipo de datos de este parámetro es INTEGER.

descripción
Describe el sistema de coordenadas explicando su aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, la información de descripción sobre el sistema de coordenadas no cambia.

El tipo de datos de este parámetro es VARCHAR(256).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_alter_coordsys. Este ejemplo utiliza un mandato CALL de DB2 para actualizar un sistema de coordenadas denominado NORTH_AMERICAN_TEST. Este mandato CALL asigna un valor de 1002 al parámetro *id_sistcoord*:

```
call db2gse.ST_alter_coordsys('NORTH_AMERICAN_TEST',NULL,NULL,1002,NULL,?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_create_coordsys" en la página 244
- "ST_drop_coordsys" en la página 256

ST_alter_srs

Utilice este procedimiento almacenado para actualizar una definición de sistema de referencia espacial en la base de datos. Cuando se procesa este procedimiento almacenado, la información acerca del sistema de referencia espacial se actualiza en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Internamente, DB2 Spatial Extender almacena los valores de coordenadas como números enteros positivos. De esta manera, durante el cálculo, se puede reducir el impacto de errores de redondeo (que dependen en gran medida del valor real para las operaciones de coma flotante). El rendimiento de las operaciones espaciales también puede mejorar de forma significativa.

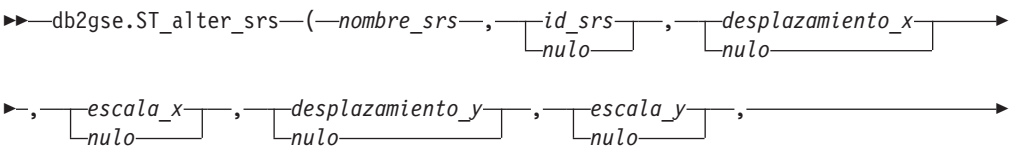
Restricción: no puede modificar un sistema de referencia espacial si una columna espacial registrada utiliza ese sistema de referencia espacial.

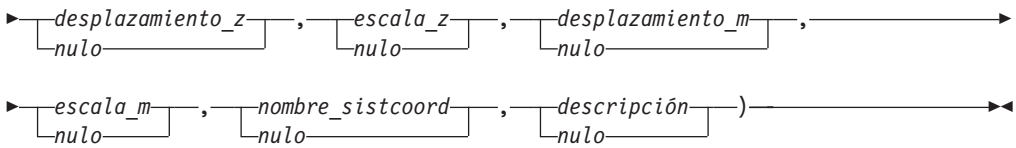
Atención: Tenga cuidado al utilizar este procedimiento almacenado. Si utiliza este procedimiento almacenado para cambiar los parámetros de desplazamiento, escala o *nombre_sistcoord* del sistema de referencia espacial, y si tiene datos espaciales existentes que están asociados con el sistema de referencia espacial, es posible que sin darse cuenta modifique los datos espaciales. Si los datos espaciales resultan afectados, el usuario es responsable de asegurar que los datos espaciales modificados sigan siendo precisos y válidos.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización SYSADM o DBADM.

Sintaxis:





Descripciones de parámetros:

nombre_srs
Identifica el sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_srs* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

id_srs
Identifica de forma exclusiva el sistema de referencia espacial. Este identificador se utiliza como parámetro de entrada para varias funciones espaciales. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el identificador numérico del sistema de referencia espacial no cambia.

El tipo de datos de este parámetro es INTEGER.

desplazamiento_x
Especifica el desplazamiento de todas las coordenadas X de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El desplazamiento se resta antes que se aplique el factor de escala *escala_x* cuando se convierten las geometrías de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. (*WKT* es texto convencional, y *WKB* es binario convencional).

El tipo de datos de este parámetro es DOUBLE.

escala_x
Especifica el factor de escala para todas las coordenadas X de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_x* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

desplazamiento_y
Especifica el desplazamiento para todas las coordenadas Y de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El desplazamiento se resta antes de aplicar el factor de escala *escala_y* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

escala_y
Especifica el factor de escala para todas las coordenadas Y de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_y* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Este factor de escala debe ser el mismo que *escala_x*.

El tipo de datos de este parámetro es DOUBLE.

desplazamiento_z
Especifica el desplazamiento para todas las coordenadas Z de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El desplazamiento se resta antes de aplicar el factor de escala *escala_z* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

escala_z
Especifica el factor de escala para todas las coordenadas Z de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_z* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

desplazamiento_m
Especifica el desplazamiento para todas las coordenadas M de las geometrías que se representan en este sistema de referencia espacial. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El desplazamiento se resta antes de aplicar el factor de escala *escala_m* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

escala_m
Especifica el factor de escala para todas las coordenadas M de las geometrías que se representan en este sistema de referencia espacial. Aunque debe

especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor para este parámetro en la definición del sistema de referencia espacial no cambia.

El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_m* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender.

El tipo de datos de este parámetro es DOUBLE.

nombre_sistcoord
Identifica de forma exclusiva el sistema de coordenadas en el que se basa este sistema de referencia espacial. El sistema de coordenadas debe estar listado en la vista ST_COORDINATE_SYSTEMS. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el sistema de coordenadas que se utiliza para este sistema de referencia espacial no cambia.

El valor de *nombre_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

descripción
Describe el sistema de referencia espacial explicando su aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, la información de descripción sobre el sistema de referencia espacial no cambia.

El tipo de datos de este parámetro es VARCHAR(256).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo se utiliza el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_alter_srs. Este ejemplo utiliza un mandato CALL de DB2 para cambiar el valor del parámetro *descripción* de un sistema de referencia espacial denominado SRSDEMO:

```
call db2gse.ST_alter_srs('SRSDEMO',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
     NULL,NULL,'SRS for GSE Demo Program: offices table',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_drop_srs" en la página 257
- "ST_create_srs" en la página 246

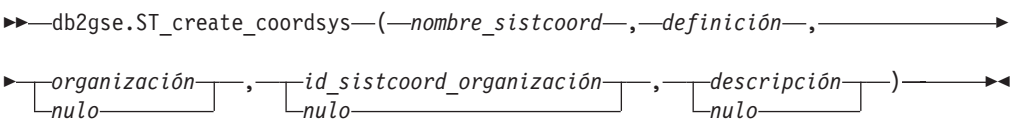
ST_create_coordsys

Utilice este procedimiento almacenado para almacenar información en la base de datos acerca del nuevo sistema de coordenadas. Cuando se procesa este procedimiento almacenado, la información acerca de este sistema de coordenadas se añade a la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización SYSADM o DBADM.

Sintaxis:



Descripciones de parámetros:

nombre_sistcoord
Identifica de forma exclusiva el sistema de coordenadas. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

definición
Define el sistema de coordenadas. Debe especificar un valor que no sea nulo para este parámetro. El proveedor que suministra el sistema de coordenadas normalmente proporciona la información para este parámetro.

El tipo de datos de este parámetro es VARCHAR(2048).

organización
Especifica la organización que ha definido el sistema de coordenadas y proporcionado la definición para la misma; por ejemplo, "European Petroleum Survey Group (EPSG)". Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si este parámetro tiene un valor nulo, el parámetro *id_sistcoord_organización* también debe tener un valor nulo. Si este parámetro no tiene un valor nulo, el parámetro *id_sistcoord_organización* no puede tener un valor nulo; en este caso,

la cambinación de los parámetros *organización* y *id_sistcoord_organización* identifica de forma exclusiva el sistema de coordenadas.

El tipo de datos de este parámetro es VARCHAR(128).

id_sistcoord_organización
Especifica un identificador numérico. La entidad que está especificada en el parámetro *organización* asigna este valor. Este valor no es necesariamente exclusivo en todo el sistema de coordenadas. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si este parámetro tiene un valor nulo, el parámetro *organización* también debe tener un valor nulo. Si este parámetro no tiene un valor nulo, el parámetro *organización* no puede tener un valor nulo; en este caso, la combinación de los parámetros *organización* y *id_sistcoord_organización* identifica de forma exclusiva el sistema de coordenadas.

El tipo de datos de este parámetro es INTEGER.

descripción
Describe el sistema de coordenadas explicando su aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se registra ninguna información de descripción sobre el sistema de coordenadas.

El tipo de datos de este parámetro es VARCHAR(256).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_create_coordsys. Este ejemplo utilizar un mandato CALL de DB2 para crear un sistema de coordenadas con los siguientes parámetros:

- parámetro *nombre_sistcoord*: NORTH_AMERICAN_TEST
- parámetro *definición*:

```
  GEOGCS["GCS_North_American_1983",
  DATUM["D_North_American_1983",
  SPHEROID["GRS_1980",6378137.0,298.257222101]],
  PRIMEM["Greenwich",0.0],
  UNIT["Degree",0.0174532925199433]]
```

- parámetro *organización*: EPSG
- parámetro *id_sistcoord_organización*: 1001
- parámetro *descripción*: Test Coordinate Systems

```
call db2gse.ST_create_coordsys('NORTH_AMERICAN_TEST',
     'GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",
     SPHEROID["GRS_1980",6378137.0,298.257222101]],
     PRIMEM["Greenwich",0.0],UNIT["Degree",
     0.0174532925199433]]','EPSG',1001,'Test Coordinate Systems',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_drop_srs" en la página 257
- "ST_alter_srs" en la página 240

ST_create_srs

Utilice estos procedimientos almacenados para crear un sistema de referencia espacial. Un sistema de referencia espacial se define por el sistema de coordenadas, la precisión y las extensiones de las coordenadas que se representan en este sistema de referencia espacial. Las extensiones son los valores de las coordenadas mínimas y máximas para las coordenadas X, Y, Z y M.

Internamente, DB2 Spatial Extender almacena los valores de coordenadas como números enteros positivos. De esta manera, durante el cálculo, se puede reducir el impacto de errores de redondeo (que dependen en gran medida del valor real para las operaciones de coma flotante). El rendimiento de las operaciones espaciales también puede mejorar de forma significativa.

Este procedimiento almacenado tiene dos variaciones:

- La primera variación toma los factores de conversión (desplazamientos y factores de escala) como parámetros de entrada.
- La segunda variación toma las extensiones y la precisión como parámetros de entrada y calcula internamente los factores de conversión.

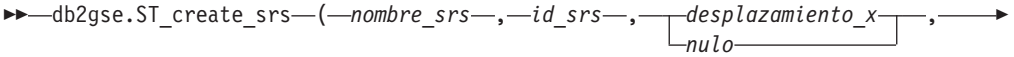
Este procedimiento almacenado sustituye a db2gse.gse_enable_sref.

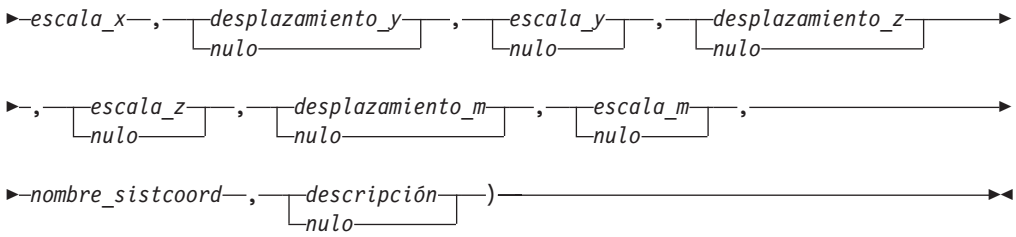
Autorización:

No se necesita ninguna.

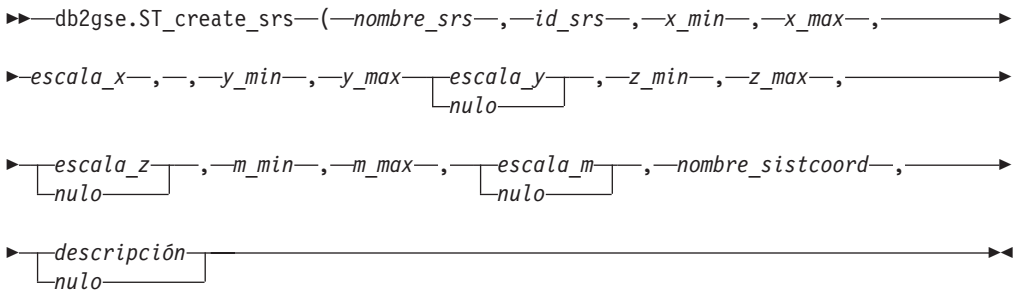
Sintaxis:

Con factores de conversión (versión 1):





Con la máxima extensión posible (versión 2):



Descripciones de parámetros:
Con factores de conversión (versión 1):

nombre_srs
Identifica el sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_srs* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

id_srs
Identifica de forma exclusiva el sistema de referencia espacial. Este identificador numérico se utiliza como parámetro de entrada para varias funciones espaciales. Debe especificar un valor que no sea nulo para este parámetro.

Para un sistema de referencia espacial geodésico, el valor *id_srs* debe estar comprendido entre 2000000318 y 2000001000. DB2 Geodetic Extender proporciona sistemas de referencia espacial geodésicos predefinidos con valores *id_srs* comprendidos entre 2000000000 y 2000000317.

El tipo de datos de este parámetro es INTEGER.

desplazamiento_x
Especifica el desplazamiento de todas las coordenadas X de las geometrías que se representan en este sistema de referencia espacial. El desplazamiento se resta antes que se aplique el factor de escala *escala_x* cuando se convierten las geometrías de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. (*WKT* es texto convencional y *WKB* es binario convencional). Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 0 (cero).

El tipo de datos de este parámetro es DOUBLE.

escala_x
Especifica el factor de escala para todas las coordenadas X de las geometrías

que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_x* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Se especifica explícitamente el valor de *desplazamiento_x* o se utiliza un valor por omisión de *desplazamiento_x* de 0. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

desplazamiento_y
Especifica el desplazamiento para todas las coordenadas Y de las geometrías que se representan en este sistema de referencia espacial. El desplazamiento se resta antes de aplicar el factor de escala *escala_y* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 0 (cero).

El tipo de datos de este parámetro es DOUBLE.

escala_y
Especifica el factor de escala para todas las coordenadas Y de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_y* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Se especifica explícitamente un valor de *desplazamiento_y* o se utiliza el valor por omisión de *desplazamiento_y* de 0. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará el valor del parámetro *escala_x*. Si especifica un valor distinto de nulo para este parámetro, el valor que especifique debe coincidir con el valor del parámetro *escala_x*.

El tipo de datos de este parámetro es DOUBLE.

desplazamiento_z
Especifica el desplazamiento para todas las coordenadas Z de las geometrías que se representan en este sistema de referencia espacial. El desplazamiento se resta antes de aplicar el factor de escala *escala_z* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 0 (cero).

El tipo de datos de este parámetro es DOUBLE.

escala_z
Especifica el factor de escala para todas las coordenadas Z de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_z* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Se especifica explícitamente el valor de *desplazamiento_z* o se utiliza un valor por omisión de *desplazamiento_z* de 0. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 1.

El tipo de datos de este parámetro es DOUBLE.

desplazamiento_m
Especifica el desplazamiento para todas las coordenadas M de las geometrías que se representan en este sistema de referencia espacial. El desplazamiento se resta antes de aplicar el factor de escala *escala_m* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 0 (cero).

El tipo de datos de este parámetro es DOUBLE.

escala_m
Especifica el factor de escala para todas las coordenadas M de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_m* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. Se especifica explícitamente el valor de *desplazamiento_m* o se utiliza un valor por omisión de *desplazamiento_m* de 0. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 1.

El tipo de datos de este parámetro es DOUBLE.

nombre_sistcoord
Identifica de forma exclusiva el sistema de coordenadas en el que se basa este sistema de referencia espacial. El sistema de coordenadas debe estar listado en la vista ST_COORDINATE_SYSTEMS. Debe proporcionar un valor que no sea nulo para este parámetro.

El valor de *nombre_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

descripción
Describe el sistema de referencia espacial explicando el propósito de su aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se registrará ninguna información de descripción.

El tipo de datos de este parámetro es VARCHAR(256).

Con la máxima extensión posible (versión 2):

nombre_srs
Identifica el sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_srs* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

id_srs
Identifica de forma exclusiva el sistema de referencia espacial. Este identificador numérico se utiliza como parámetro de entrada para varias funciones espaciales. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es INTEGER.

x_min
Especifica el valor mínimo posible de coordenada X para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

x_max
Especifica el valor máximo posible de coordenada X para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

En función del valor de *escala_x*, el valor que se muestra en la vista DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS puede ser mayor que el valor especificado aquí. El valor de la vista es correcto.

El tipo de datos de este parámetro es DOUBLE.

escala_x
Especifica el factor de escala para todas las coordenadas X de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_x* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. El cálculo del desplazamiento *desplazamiento_x* se basa en el valor *x_min*. Debe proporcionar un valor que no sea nulo para este parámetro.

Si se especifican los parámetros *escala_x* y *escala_y*, los valores deben coincidir.

El tipo de datos de este parámetro es DOUBLE.

y_min
Especifica el valor mínimo posible de coordenada Y para todas las geometrías que utilizan este sistema de referencia espacial. Debe proporcionar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

y_max
Especifica el valor máximo posible de coordenada Y para todas las geometrías que utilizan este sistema de referencia espacial. Debe proporcionar un valor que no sea nulo para este parámetro.

En función del valor de *escala_y*, el valor que se muestra en la vista DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS puede ser mayor que el valor especificado aquí. El valor de la vista es correcto.

El tipo de datos de este parámetro es DOUBLE.

escala_y
Especifica el factor de escala para todas las coordenadas Y de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_y* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. El cálculo del desplazamiento *desplazamiento_y* se basa en el valor *y_min*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará el valor del parámetro *escala_x*. Si se especifican los parámetros *escala_y* y *escala_x*, los valores deben coincidir.

El tipo de datos de este parámetro es DOUBLE.

z_min
Especifica el valor mínimo posible de coordenadas Z para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

z_max
Especifica el valor mínimo posible de coordenada Z para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

En función del valor de *escala_z*, el valor que se muestra en la vista DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS puede ser mayor que el valor especificado aquí. El valor de la vista es correcto.

El tipo de datos de este parámetro es DOUBLE.

escala_z
Especifica el factor de escala para todas las coordenadas Z de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_z* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. El cálculo del desplazamiento *desplazamiento_z* se basa en el valor *z_min*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 1.

El tipo de datos de este parámetro es DOUBLE.

m_min
Especifica el valor mínimo posible de coordenada M para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

m_max
Especifica el valor máximo posible de coordenada M para todas las geometrías que utilizan este sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

En función del valor de *escala_m*, el valor que se muestra en la vista DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS puede ser mayor que el valor especificado aquí. El valor de la vista es correcto.

El tipo de datos de este parámetro es DOUBLE.

escala_m
Especifica el factor de escala para todas las coordenadas M de las geometrías que se representan en este sistema de referencia espacial. El factor de escala se aplica (multiplicación) después de restar el desplazamiento *desplazamiento_m* cuando las geometrías se convierten de representaciones externas (WKT, WKB, forma) a la representación interna de DB2 Spatial Extender. El cálculo del desplazamiento *desplazamiento_m* se basa en el valor *m_min*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 1.

El tipo de datos de este parámetro es DOUBLE.

nombre_sistcoord
Identifica de forma exclusiva el sistema de coordenadas en el que se basa este

sistema de referencia espacial. El sistema de coordenadas debe estar listado en la vista ST_COORDINATE_SYSTEMS. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

descripción
Describe el sistema de referencia espacial explicando el propósito de su aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se registrará ninguna información de descripción.

El tipo de datos de este parámetro es VARCHAR(256).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_create_srs. Este ejemplo utiliza un mandato CALL de DB2 para crear un sistema de referencia espacial denominado SRSDEMO con los siguientes valores de parámetros:

- *id_srs*: 1000000
- *desplazamiento_x*: -180
- *escala_x*: 1000000
- *desplazamiento_y*: -90
- *escala_y*: 1000000

```
call db2gse.ST_create_srs('SRSDEMO',1000000,
                    -180,1000000, -90, 1000000,
                    0, 1, 0, 1,'NORTH_AMERICAN',
                    'SRS for GSE Demo Program: customer table',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Conceptos relacionados:

- "Sistemas de referencia espacial" en la página 68

Tareas relacionadas:

- "Creación de un sistema de referencia espacial" en la página 76

ST_disable_autogeocoding

Utilice este procedimiento almacenado para especificar que DB2 Spatial Extender deje de sincronizar una columna geocodificada con su columna o columnas de geocodificación asociadas. Se utiliza una *columna de geocodificación* como dato de entrada en el geocodificador.

Este procedimiento almacenado sustituye a db2gse.gse_disable_autogc.

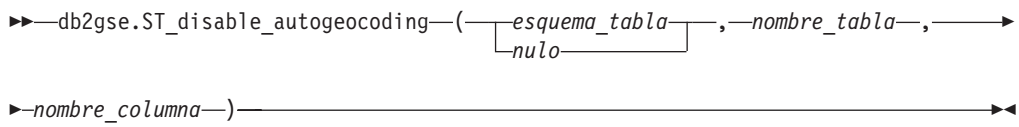
Autorización:

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener una de las autorizaciones o uno de los privilegios siguientes:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla en la que están definidos los desencadenantes que se van a descartar
- Privilegio CONTROL sobre esta tabla
- Privilegios ALTER y UPDATE sobre esta tabla

Nota: Para los privilegios CONTROL y ALTER, debe tener autorización DROPIN sobre el esquema DB2GSE.

Sintaxis:



Descripciones de parámetros:

esquema_tabla
Especifica el esquema al que pertenece la tabla o vista que está especificada en el parámetro *nombre_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_tabla
Especifica el nombre no calificado de la tabla en la que están definidos los desencadenantes que desea descartar. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_columna
Especifica la columna geocodificada que es mantenida por los desencadenantes que desea descartar. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_columna* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_disable_autogeocoding. Este ejemplo utiliza un mandato CALL de DB2 para inhabilitar la geocodificación automática en la columna LOCATION de la tabla denominada CUSTOMERS:

```
call db2gse.ST_disable_autogeocoding(NULL,'CUSTOMERS','LOCATION',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- “ST_enable_autogeocoding” en la página 258
- “ST_setup_geocoding” en la página 286

ST_disable_db

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

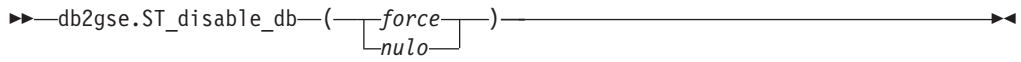
Este procedimiento almacenado le ayuda a solucionar problemas o temas que surjan tras habilitar la base de datos para operaciones espaciales. Por ejemplo, puede habilitar una base de datos para operaciones espaciales y a continuación decidir utilizar en su lugar otra base de datos con DB2 Spatial Extender. Si no ha definido ninguna columna espacial ni importado datos espaciales, puede invocar a este procedimiento almacenado para eliminar todos los recursos espaciales de la primera base de datos. Debido a esta interdependencia entre las columnas espaciales y las definiciones de tipos, no puede descartar las definiciones de tipos cuando existan columnas de estos tipos. Si ya ha definido las columnas espaciales pero todavía desea inhabilitar una base de datos para operaciones espaciales, debe especificar un valor distinto de 0 (cero) para el parámetro *force* para eliminar todos los recursos espaciales de la base de datos que no tienen otras dependencias en ellos.

Este procedimiento almacenado sustituye a db2gse.gse_disable_db.

Autorización:

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos de la que se van a eliminar los recursos de DB2 Spatial Extender.

Sintaxis:



Descripciones de parámetros:

force
Especifica que desea inhabilitar una base de datos para operaciones espaciales, aunque puede tener objetos de base de datos que sean dependientes de los tipos espaciales o de las funciones espaciales. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si especifica un valor distinto de 0 (cero) o nulo para el parámetro *force*, la base de datos se inhabilita y se eliminan todos los recursos de DB2 Spatial Extender (si es posible). Si especifica 0 (cero) o nulo, la base de datos no se inhabilita si los objetos de base de datos son dependientes de tipos espaciales o de funciones espaciales. Los objetos de base de datos que pueden tener este tipo de dependencias incluyen tablas, vistas, restricciones, desencadenantes, columnas generadas, métodos, funciones, procedimientos y otros tipos de datos (subtipos o tipos estructurados con un atributo espacial).

El tipo de datos de este parámetro es SMALLINT.

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_disable_db. Este ejemplo utiliza un mandato CALL de DB2 para inhabilitar la base de datos para operaciones espaciales, con un valor del parámetro *force* de 1:

```
call db2gse.ST_disable_db(1,?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_alter_coordsys" en la página 238
- "ST_create_coordsys" en la página 244

ST_drop_coordsys

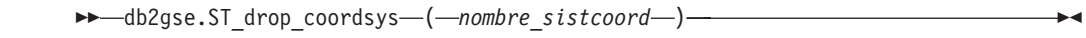
Utilice este procedimiento almacenado para suprimir información acerca de un sistema de coordenadas de la base de datos. Cuando se procesa este procedimiento almacenado, la información sobre el sistema de coordenadas se elimina de la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Restricción: No puede descartar un sistema de coordenadas en el que se basa un sistema de referencia espacial.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización SYSADM o DBADM.

Sintaxis:



Descripciones de parámetros:

nombre_sistcoord
Identifica de forma exclusiva el sistema de coordenadas. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_sistcoord* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_drop_coordsys. Este ejemplo utiliza un mandato CALL de DB2 para suprimir de la base de datos un sistema de coordenadas denominado NORTH_AMERICAN_TEST:

```
call db2gse.ST_drop_coordsys('NORTH_AMERICAN_TEST',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

ST_drop_srs

Utilice este procedimiento almacenado para descartar un sistema de referencia espacial. Cuando se procesa este procedimiento almacenado, la información sobre el sistema de referencia espacial se elimina de la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Restricción: No puede descartar un sistema de referencia espacial si una columna espacial que utiliza dicho sistema de referencia espacial está registrada.

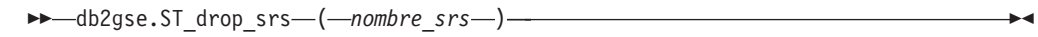
Importante: Tenga precaución al utilizar este procedimiento almacenado. Si utiliza este procedimiento almacenado para descartar un sistema de referencia espacial y si algunos datos espaciales están asociados con este sistema de referencia espacial, ya no puede realizar operaciones espaciales sobre los datos espaciales.

Este procedimiento almacenado sustituye a db2gse.gse_disable_sref.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización SYSADM o DBADM.

Sintaxis:



Descripciones de parámetros:

nombre_srs
Identifica el sistema de referencia espacial. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_srs* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_drop_srs. Este ejemplo utiliza un mandato CALL de DB2 para suprimir un sistema de referencia espacial denominado SRSDEMO:

```
call db2gse.ST_drop_srs('SRSDEMO',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_create_srs" en la página 246
- "ST_alter_srs" en la página 240

ST_enable_autogeocoding

Utilice este procedimiento almacenado para especificar que DB2 Spatial Extender sincronice una columna geocodificada con su columna o sus columnas de geocodificación asociadas. Una *columna de geocodificación* se utiliza como dato de entrada en el geocodificador. Cada vez que se insertan o actualizan valores en la

columna o columnas de geocodificación, se activan desencadenantes. Estos desencadenantes invocan al geocodificador asociado para geocodificar los valores insertados o actualizados y para colocar los datos resultantes en la columna geocodificada.

Restricción: Puede habilitar la geocodificación automática sólo en tablas para las que se pueden crear desencadenantes de inserción y actualización. En consecuencia, no puede habilitar la geocodificación automática en vistas ni apodos.

Requisito necesario: Antes de habilitar la geocodificación automática, debe realizar el paso de configuración de la geocodificación invocando al procedimiento ST_setup_geocoding. El paso de configuración de la geocodificación especifica el geocodificador y los valores de los parámetros de geocodificación. Además, identifica las columnas de geocodificación que se van a sincronizar con las columnas geocodificadas.

Este procedimiento almacenado sustituye a db2gse.gse_enable_autogc.

Autorización:

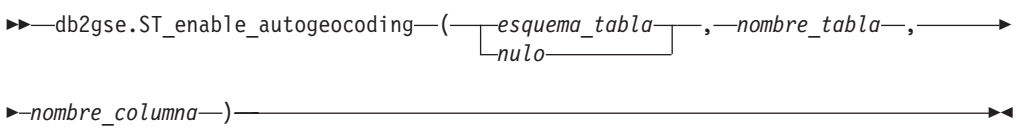
El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener una de las autorizaciones o uno de los privilegios siguientes:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla en la que están definidos los desencadenantes que ha creado este procedimiento almacenado
- Privilegio CONTROL sobre la tabla
- Privilegio ALTER sobre la tabla

Si el ID de autorización de la sentencia no tiene autorización SYSADM ni DBADM, los privilegios que tenga el ID de autorización de la sentencia (sin tener en cuenta privilegios PUBLIC o de grupo) deben incluir todos los privilegios siguientes mientras exista el desencadenante:

- Privilegio SELECT sobre la tabla en la que está habilitada la geocodificación automática
- Privilegios necesarios para evaluar las expresiones de SQL que se especifican para los parámetros en la configuración de geocodificación

Sintaxis:



Descripciones de parámetros:

esquema_tabla
Identifica el esquema al que pertenece la tabla que está especificada en el parámetro *nombre_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla.

El valor de *esquema_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_tabla
Especifica el nombre no calificado de la tabla que contiene la columna en la que se insertarán o se actualizarán los datos geocodificados. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_columna
Identifica la columna en la que se insertarán o actualizarán los datos geocodificados. Esta columna se denomina columna geocodificada. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos para invocar al procedimiento almacenado ST_enable_autogeocoding. Este ejemplo utiliza un mandato CALL de DB2 para habilitar la geocodificación en la columna LOCATION de la tabla denominada CUSTOMERS:

```
call db2gse.ST_enable_autogeocoding(NULL,'CUSTOMERS','LOCATION',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_setup_geocoding" en la página 286

ST_enable_db

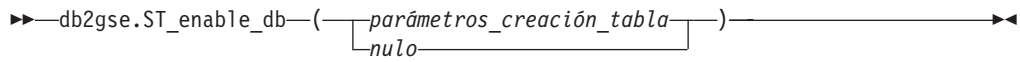
Utilice este procedimiento almacenado para proporcionar a la base de datos los recursos que necesita para almacenar datos espaciales y para dar soporte a operaciones espaciales. Estos recursos incluyen tipos de datos espaciales, tipos de índices espaciales, vistas de catálogo, funciones proporcionadas y otros procedimientos almacenados.

Este procedimiento almacenado sustituye a db2gse.gse_enable_db.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos que se está habilitando.

Sintaxis:



Descripciones de parámetros:

parámetros_creación_tabla
Especifica las opciones que se van a añadir a las sentencias CREATE TABLE para las tablas de catálogo de DB2 Spatial Extender. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se añade ninguna opción a las sentencias CREATE TABLE.

Para especificar estas opciones, utilice la sintaxis de la sentencia DB2 CREATE TABLE. Por ejemplo, para especificar un espacio de tabla en el que se crearán las tablas, utilice:

```
IN Nombre_ET INDEX IN Nombre_índice_ET
```

El tipo de datos de este parámetro es VARCHAR(32K).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

El siguiente ejemplo muestra cómo utilizar la CLI (Call Level Interface) para invocar al procedimiento almacenado ST_enable_db:

```
SQLHANDLE  henv;
  SQLHANDLE  hdbc;
  SQLHANDLE  hstmt;
  SQLCHAR    uid[MAX_UID_LENGTH + 1];
  SQLCHAR    pwd[MAX_PWD_LENGTH + 1];
  SQLINTEGER ind[3];
  SQLINTEGER msg_code = 0;
  char       msg_text[1024] = "";
  SQLRETURN  rc;
  char       *table_creation_parameters = NULL;

  /* Asignar descriptor de entorno */
  rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);

  /* Asignar descriptor de base de datos */
  rc = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);

  /* Establecer una conexión con la base de datos "testdb" */
  rc = SQLConnect(hdbc, (SQLCHAR *)"testdb", SQL_NTS, (SQLCHAR *)uid,SQL_NTS,
                  (SQLCHAR *)pwd, SQL_NTS);


  /* Asignar descriptor de sentencia */
  rc = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt) ;

  /* Asociar sentencia de SQL para llamar al procedimiento          */
  /* almacenado ST_enable_db con el descriptor de sentencia y       */
  /* enviar la sentencia a DBMS para estar preparado                */
  rc = SQLPrepare(hstmt, "call db2gse!ST_enable_db(?,?,?)", SQL_NTS);

  /* Vincular el primer marcador de parámetros de la sentencia         */
  /* de llamada de SQL, el parámetro de entrada para los parámetros    */
  /* de creación de tablas, con la variable table_creation_parameters. */
   ind[0] = SQL_NULL_DATA;
  rc = SQLBindParameter(hstmt, 1, SQL_PARAM_OUTPUT, SQL_C_CHAR,
       SQL_VARCHAR, 255, 0, table_creation_parameters, 256, &ind[0]);

  /* Vincular el segundo marcador de parámetros de la sentencia   */
  /* de llamada de SQL, el parámetro de salida del código de      */
  /* mensaje devuelto, con la variable msg_code.                  */
  ind[1] = 0;
  rc = SQLBindParameter(hstmt, 2, SQL_PARAM_OUTPUT, SQL_C_LONG,
                        SQL_INTEGER, 0, 0, &msg_code, 4, &ind[1]);

  /* Vincular el tercer marcador de parámetro de la sentencia     */
  /* SQL, el parámetro de salida del texto del mensaje devuelto,  */
  /* con la variable msg_text.                                    */
  ind[2] = 0;
  rc = SQLBindParameter(hstmt, 3, SQL_PARAM_OUTPUT, SQL_C_CHAR,
                        SQL_VARCHAR, (sizeof(msg_text)-1), 0, msg_text,
                         sizeof(msg_text), &ind[2]);
  rc = SQLExecute(hstmt);
```

Información relacionada:

- "ST_disable_db" en la página 254

ST_export_shape

Utilice este procedimiento almacenado para exportar una columna espacial y su tabla asociada a un archivo de formas.

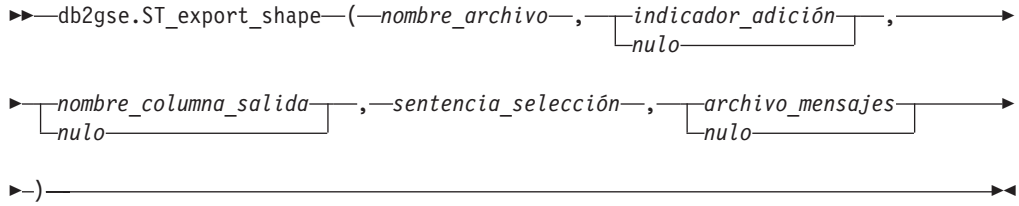
Este procedimiento almacenado sustituye a db2gse.gse_export_shape.

Autorización:

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener los privilegios necesarios para ejecutar satisfactoriamente la sentencia SELECT desde la que se van a exportar los datos.

El procedimiento almacenado, que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios en el servidor para crear los archivos de formas y escribir en los mismos.

Sintaxis:



Descripciones de parámetros:

nombre_archivo
Especifica la vía de acceso completa de un archivo de formas al que se van a exportar los datos especificados. Debe especificar un valor que no sea nulo para este parámetro.

Puede utilizar el procedimiento almacenado ST_export_shape para exportar un nuevo archivo o para exportar a un archivo existente añadiendo los datos exportados al mismo:

- Si está exportando a un nuevo archivo, puede especificar la extensión opcional de archivo como .shp o .SHP. Si especifica .shp o .SHP para la extensión del archivo, DB2 Spatial Extender crea el archivo con el valor *nombre_archivo* especificado. Si no especifica la extensión opcional de archivo, DB2 Spatial Extender crea el archivo que tiene el nombre del valor *nombre_archivo* que el usuario especifica con una extensión .shp.
- Si está exportando datos añadiendo datos a un archivo existente, DB2 Spatial Extender primero busca una coincidencia exacta del nombre que especifica para el parámetro *nombre_archivo*. Si DB2 Spatial Extender no encuentra una coincidencia exacta, busca primero un archivo con la extensión .shp y, a continuación, un archivo con la extensión .SHP.

Si el valor del parámetro *indicador_adición* indica que no está realizando adiciones a un archivo existente, pero que el archivo nombrado en el parámetro *nombre_archivo* ya existe, DB2 Spatial Extender devuelve un error y sobrescribe el archivo.

Consulte el apartado “Notas de utilización” en la página 265 para ver una lista de los archivos que se escriben en el servidor. El procedimiento almacenado,

que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios en el servidor para crear y grabar archivos.

El tipo de datos de este parámetro es VARCHAR(256).

indicador_adición
Indica si los datos que se van a exportar se van a añadir a un archivo de formas existente. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Indique si desea realizar adiciones a un archivo de formas existente de la forma siguiente:

- Si desea añadir datos a un archivo de formas existente, especifique cualquier valor distinto de 0 (cero) y nulo. En este caso, la estructura del archivo debe coincidir con los datos exportados; en caso contrario, se devuelve un error.
- Si desea exportar a un nuevo archivo, especifique 0 (cero) o nulo. En este caso, DB2 Spatial Extender no sobrescribe ningún archivo existente.

El tipo de datos de este parámetro es SMALLINT.

nombres_columnas_salida
Especifica uno o varios nombres de columna (separados por comas) que se utilizarán para columnas no espaciales en el archivo dBASE de salida. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizan los nombres que se derivan de la sentencia SELECT.

Si especifica este parámetro pero no coloca los nombres de columna entre comillas dobles, los nombres de columna se convertirán a mayúsculas. El número de columnas especificadas debe coincidir con el número de columnas que se devuelven de la sentencia SELECT, según se especifica en el parámetro *sentencia_selección*, excluyendo la columna espacial.

El tipo de datos de este parámetro es VARCHAR(32K).

sentencia_selección
Especifica el subelemento que devuelve los datos que se van a exportar. El subelemento debe hacer referencia exactamente a una columna espacial y a cualquier número de columnas de atributos. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es VARCHAR(32K).

archivo_mensajes
Especifica el nombre de la vía de acceso completa del archivo (en el servidor) que contendrá mensajes sobre la operación de exportación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se creará ningún archivo para mensajes de DB2 Spatial Extender.

Los mensajes que se envían a este archivo de mensajes pueden ser:

- Mensajes informativos, como, por ejemplo, un resumen de la operación de exportación
- Mensajes de error para datos que no se han podido exportar, por ejemplo, debido a sistemas distintos de coordenadas

El procedimiento almacenado, que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios para crear el archivo.

El tipo de datos de este parámetro es VARCHAR(256).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Notas de uso:

Puede exportar sólo una columna espacial cada vez.

El procedimiento almacenado ST_export_shape crea y graba los cuatro archivos siguientes:

- El archivo principal de formas (extensión .shp).
- El archivo de índices de formas (extensión .shx).
- Un archivo dBASE que contiene datos para columnas no espaciales (extensión .dbf). Este archivo se crea solamente si las columnas de atributos realmente necesitan ser exportadas
- Un archivo de proyección que especifica el sistema de coordenadas que está asociado con los datos espaciales, si el sistema de coordenadas no es igual a ″SIN ESPECIFICAR″ (extensión .prj). El sistema de coordenadas se obtiene del primer registro espacial. Se produce un error si los registros subsiguientes tienen sistemas de coordenadas diferentes.

La tabla siguiente describe cómo los tipos de datos de DB2 se almacenan en archivos de atributos dBASE. Los demás tipos de datos de DB2 no están soportados.

Tabla 29. Almacenamiento de tipos de datos de DB2 en archivos de atributos

Tipo SQL	Tipo .dbf	Tipo .dbf	Decimales .dbf	Comentarios
SMALLINT	N	6	0	
INTEGER	N	11	0	
BIGINT	N	20	0	
DECIMAL	N	precision+2	scale	
REAL FLOAT(1) hasta FLOAT(24)	F	14	6	
DOUBLE FLOAT(25) hasta FLOAT(53)	F	19	9	

Tabla 29. Almacenamiento de tipos de datos de DB2 en archivos de atributos (continuación)

Tipo SQL	Tipo .dbf	Tipo .dbf	Decimales .dbf	Comentarios
CHARACTER, VARCHAR, LONG VARCHAR y DATALINK	C	*len*	0	longitud ≤ 255
DATE	D	8	0	
TIME	C	8	0	
TIMESTAMP	C	26	0	

Se da soporte a todos los sinónimos para los tipos de datos y tipos diferenciados que están basados en los tipos listados en la tabla precedente.

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_export_shape. Este ejemplo utiliza un mandato CALL de DB2 para exportar todas las filas de la tabla CUSTOMERS a un archivo de formas que se creará y se denominará `/tmp/export_file`:

```
call db2gse.ST_export_shape('/tmp/export_file',0,NULL,
     'select * from customers','/tmp/export_msg',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_import_shape" en la página 266

ST_import_shape

Utilice este procedimiento almacenado para importar un archivo de formas a una base de datos que está habilitada para operaciones espaciales. El procedimiento almacenado puede funcionar de cualquiera de estas dos maneras, en función del parámetro *indicador_creación_tabla*:

- DB2 Spatial Extender puede crear una tabla que tiene una columna espacial y columnas de atributos, y a continuación puede cargar las columnas de la tabla con los datos de archivo.
- En caso contrario, los datos de formas y atributos se pueden cargar en una tabla existente que tenga una columna espacial y columnas de atributos que coincidan con los datos del archivo.

Este procedimiento almacenado sustituye a db2gse.gse_import_shape.

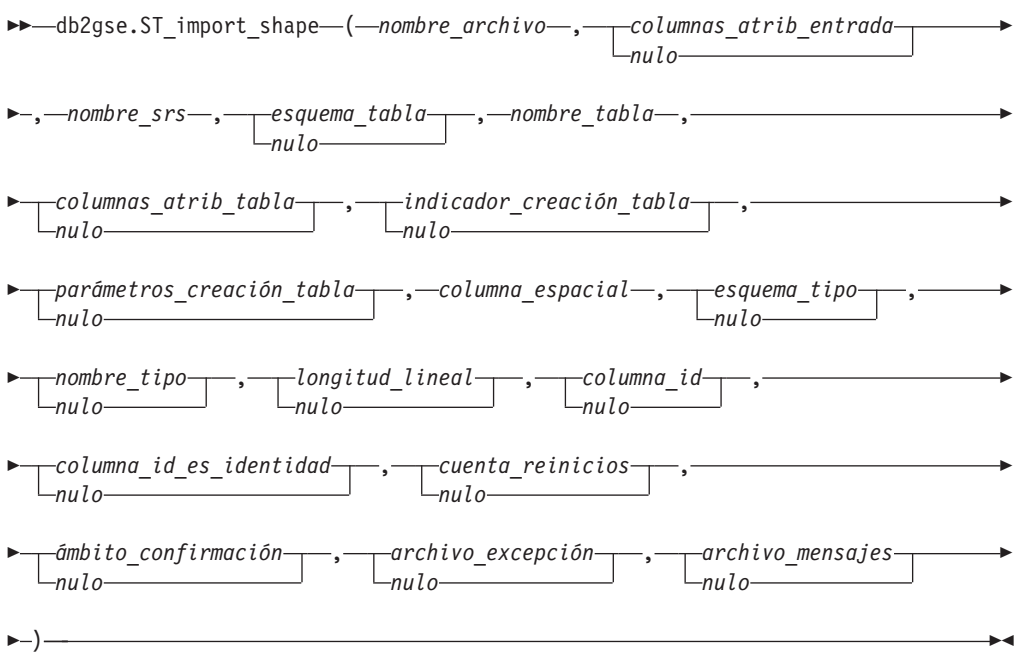
Autorización:

El propietario de la instancia de DB2 debe tener los privilegios necesarios en el servidor para leer los archivos de entrada y opcionalmente para escribir en archivos de errores. Los requisitos de autorizaciones adicionales varían en función de si está importando a una tabla existente o a una nueva tabla.

- **Cuando importe a una tabla existente**, el ID de usuario utilizado para invocar a este procedimiento almacenado debe tener una de las siguientes autorizaciones o uno de los siguientes privilegios:
  - SYSADM o DBADM
  - Privilegio CONTROL sobre la tabla o la vista
  - Privilegio INSERT y SELECT en la tabla o vista
- **Cuando importe a una nueva tabla**, el ID de usuario utilizado para invocar a este procedimiento almacenado debe tener una de las siguientes autorizaciones o uno de los siguientes privilegios:
  - SYSADM o DBADM
  - Autorización CREATETAB sobre la base de datos

  El ID de usuario debe tener también una de las autorizaciones siguientes:
  - Autorización IMPLICIT_SCHEMA sobre la base de datos, si el nombre de esquema de la tabla no existe
  - Privilegio CREATEIN sobre el esquema, si el esquema de la tabla existe

Sintaxis:



Descripciones de parámetros:

nombre_archivo
Especifica el nombre de la vía de acceso completa del archivo de formas que se va a importar. Debe especificar un valor que no sea nulo para este parámetro.

Si especifica la extensión opcional de archivo, especifique .shp o .SHP. DB2 Spatial Extender primero busca una coincidencia exacta del nombre de archivo especificado. Si DB2 Spatial Extender no encuentra una coincidencia exacta, busca primero un archivo con la extensión .shp y, a continuación, un archivo con la extensión .SHP.

Consulte el apartado “Notas de utilización” en la página 273 para ver una lista de los archivos necesarios, que deben estar ubicados en el servidor. El

procedimiento almacenado, que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios en el servidor para leer los archivos.

El tipo de datos de este parámetro es VARCHAR(256).

columnas_atrib_entrada
Especifica una lista de columnas de atributos para importar desde el archivo dBASE. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se importan todas las columnas. Si el archivo dBASE no existe, este parámetro debe ser una cadena vacía de caracteres o tener un valor nulo.

Para especificar un valor que no sea nulo para este parámetro, utilice una de las especificaciones siguientes:

- **Listar los nombres de columnas de atributos.** El ejemplo siguiente muestra cómo especificar una lista de los nombres de las columnas de atributos que se van a importar desde el archivo dBASE:

  ```
  N(COLUMN1,COLUMN5,COLUMN3,COLUMN7)
  ```

  Si un nombre de columna no está entre comillas dobles, se convierte a mayúsculas. Cada nombre de la lista debe estar separado por una coma. Los nombres resultantes deben coincidir exactamente con los nombres de columna en el archivo dBASE.
- **Listar los números de columnas de atributos.** El ejemplo siguiente muestra cómo especificar una lista de los números de las columnas de atributos que se van a importar desde el archivo dBASE:

  ```
  P(1,5,3,7)
  ```

  Las columnas están numeradas empezando por 1. Cada número de la lista debe estar separado por una coma.
- **Indicar que no se van a importar datos de atributos.** Especifique ″″, que es una cadena vacía de caracteres que especifica explícitamente que DB2 Spatial Extender *no* va a importar datos de atributos.

El tipo de datos de este parámetro es VARCHAR(32K).

nombre_srs
Identifica el sistema de referencia espacial que se va a utilizar para las geometrías que se importan en la columna espacial. Debe especificar un valor que no sea nulo para este parámetro.

La columna espacial no estará registrada. El sistema de referencia espacial (SRS) debe existir antes de importar los datos. El proceso de importación no crea implícitamente el SRS, pero compara el sistema de coordenadas del SRS con el sistema de coordenadas que está especificado en el archivo .prj (si está disponible con el archivo de formas). El proceso de importación también verifica que las extensiones de los datos en el archivo de formas pueden estar representadas en dicho sistema de referencia espacial. Es decir, el proceso de importación verifica que las extensiones estén dentro de las coordenadas X, Y, Z y M mínimas y máximas posibles del SRS.

El valor de *nombre_srs* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

esquema_tabla
Especifica el esquema al que pertenece la tabla que está especificada en el parámetro *nombre_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_tabla
Especifica el nombre no calificado de la tabla en la que se cargará el archivo de formas importado. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

columnas_atrib_tabla
Especifica los nombres de columna de la tabla donde se almacenarán los datos de atributos del archivo dBASE. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizan los nombres de las columnas en el archivo dBASE.

Si se especifica este parámetro, el número de nombres debe coincidir con el número de columnas que se importan desde el archivo dBASE. Si existe la tabla, las definiciones de columna deben coincidir con los datos entrantes. Consulte el apartado “Notas de utilización” en la página 273 para ver una explicación de cómo los tipos de datos de atributos se correlacionan con tipos de datos de DB2.

El tipo de datos de este parámetro es VARCHAR(32K).

indicador_creación_tabla
Especifica si el proceso de importación creará una nueva tabla. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo o cualquier otro valor distinto de 0 (cero), se creará una nueva tabla. (Si la tabla ya existe, se devuelve un error.) Si este parámetro es 0 (cero), no se creará ninguna tabla y ya tabla ya deberá existir.

El tipo de datos de este parámetro es INTEGER.

parámetros_creación_tabla
Especifica las opciones que se añadirán a la sentencia CREATE TABLE que crea una tabla en la que se importarán los datos. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se añadirá ninguna opción a la sentencia CREATE TABLE.

Para especificar las opciones CREATE TABLE, utilice la sintaxis de la sentencia CREATE TABLE de DB2. Por ejemplo, para especificar un espacio de tabla para crear tablas, especifique:

```
IN Nombre_ET INDEX IN Nombre_índice_ET LONG IN Nombre_ET_largo
```

El tipo de datos de este parámetro es VARCHAR(32K).

columna_espacial
Nombre de la columna espacial de la tabla en la que se cargarán los datos de formas. Debe especificar un valor que no sea nulo para este parámetro.

Para una nueva tabla, este parámetro especifica el nombre de la nueva columna espacial que se creará. En caso contrario, este parámetro especifica el nombre de una columna espacial existente en la tabla.

El valor de *columna_espacial* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

esquema_tipo
Especifica el nombre de esquema del tipo de datos espaciales (especificado por el parámetro *nombre_tipo*) que se utilizará al crear una columna espacial en una nueva tabla. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de DB2GSE.

El valor de *esquema_tipo* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_tipo
Especifica el tipo de datos que se utilizará para los valores espaciales. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el tipo de datos está determinado por el archivo de formas y pertenece a uno de los tipos siguientes:

- ST_Point
- ST_MultiPoint
- ST_MultiLineString
- ST_MultiPolygon

Tenga en cuenta que los archivos de formas, por definición, permiten la distinción sólo entre puntos y multipuntos, pero no entre polígonos y multipolígonos ni entre cadenas lineales y multilíneas.

Si está importando a una tabla que no existe aún, este tipo de datos también se utiliza para el tipo de datos de la columna espacial. En este caso, el tipo de datos también puede ser un supertipo de datos de ST_Point, ST_MultiPoint, ST_MultiLineString o ST_MultiPolygon.

El valor de *nombre_tipo* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

longitud_lineal
Especifica, para una nueva tabla, el máximo número de bytes que se asignarán para la columna espacial dentro de la tabla. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se utilizará ninguna opción INLINE LENGTH explícita en la sentencia CREATE TABLE y no se utilizará implícitamente ningún valor por omisión de DB2.

Los registros espaciales que exceden este tamaño se almacenan separadamente en el espacio de tabla de LOB, al que puede ser más lento acceder.

Los tamaños típicos que son necesarios para varios tipos espaciales son los siguientes:

- **Un punto:** 292.
- **Multipunto, línea o polígono:** Un valor lo más alto posible. Tenga en cuenta el número total de bytes en una fila no puede exceder el límite para el tamaño de página del espacio de tabla para el que la tabla se ha creado.

Consulte la documentación de DB2 acerca de la sentencia CREATE TABLE de SQL para ver una descripción completa de este valor. Vea también el programa de utilidad db2dart para determinar el número de geometrías en línea para tablas existentes y la posibilidad de modificar la longitud lineal.

El tipo de datos de este parámetro es INTEGER.

columna_id
Especifica la columna que se creará para contener un número único para cada fila de datos. (Las herramientas ESRI requieren una columna denominada SE_ROW_ID.) Los valores exclusivos para esta columna se generan automáticamente durante el proceso de importación. Aunque debe especificar un valor para este parámetro, el valor debe ser nulo si no existe ninguna columna (con un ID exclusivo en cada fila) en la tabla o si no está añadiendo una columna de este tipo a una tabla creada recientemente. Si este parámetro tiene un valor nulo, no se creará ni llenará ninguna columna con números exclusivos.

Restricción: No puede especificar un nombre *columna_id* que coincida con el nombre de alguna columna en el archivo dBASE.

Los requisitos y efecto de este parámetro dependen de si la tabla ya existe.

- **Para una tabla existente**, el tipo de datos del parámetro *columna_id* puede ser cualquier tipo de número entero (INTEGER, SMALLINT o BIGINT).
- **Para una nueva tabla que se creará**, la columna se añade a la tabla cuando el procedimiento almacenado la crea. La columna se definirá de la forma siguiente:

  ```
  INTEGER NOT NULL PRIMARY KEY
  ```

  Si el valor del parámetro *columna_id_es_identidad* no es nulo ni es 0 (cero), la definición se expande de la forma siguiente:

  ```
  INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY
  ( START WITH 1 INCREMENT BY 1 )
  ```

EL valor de *columna_id* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

columna_id_es_identidad
Indica si la *columna_id* especificada se creará utilizando la cláusula IDENTITY. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro es 0 (cero) o nulo, la columna no se creará como la columna de identidad. Si el parámetro es cualquier valor distinto de 0 o de nulo, la columna se creará como la columna de identidad. Este parámetro se ignora para tablas que ya existen.

El tipo de datos de este parámetro es SMALLINT.

cuenta_reinicios
Especifica que se debe iniciar una operación de importación en el registro *n* + 1. Se saltan los *n* primeros registros. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se importarán todos los registros (empezando con el número de registro 1).

El tipo de datos de este parámetro es INTEGER.

ámbito_confirmación
Especifica que se realizará un COMMIT después de que se importen *n* registros como mínimo. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se utilizará un valor de 0 (cero) y no se confirmará ningún registro.

El tipo de datos de este parámetro es INTEGER.

archivo_excepción
Especifica la vía de acceso completa de un archivo de formas en el que están almacenados los datos de formas que no se han podido importar. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si el parámetro tiene un valor nulo, no se creará ningún archivo.

Si especifica un valor para el parámetro e incluye la extensión opcional de archivo, especifique .shp o .SHP. Si la extensión del archivo tiene un valor nulo, se añadirá una extensión .shp.

El archivo de excepciones mantiene el bloque completo de filas para las que una única sentencia de insertar no ha sido satisfactoria. Por ejemplo, supongamos que una fila no se puede importar porque los datos de formas están codificados de forma incorrecta. Una única sentencia de insertar intenta importar 20 filas, incluyendo la que está en error. Debido al problema con la fila sencilla, todo el bloque de 20 filas se graba en el archivo de excepciones.

Los registros sólo se graban en el archivo de excepciones cuando estos registros se pueden identificar correctamente, como es el caso cuando el tipo de registro de forma no es válido. Algunos tipos de corrupciones en los datos de formas (archivos .shp) y el índice de formas (archivos .shx) no permiten identificar los registros adecuados. En este caso, no se graba ningún registro en el archivo de excepciones y se emite un mensaje de error para informar del problema.

Si especifica un valor para este parámetro, se crearán cuatro archivos en el servidor. Consulte el apartado "Notas de utilización" en la página 273 para ver una explicación de estos archivos. El procedimiento almacenado, que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios en el servidor para crear los archivos. Si los archivos ya existen, el procedimiento almacenado devuelve un error.

El tipo de datos de este parámetro es VARCHAR(256).

archivo_mensajes
Especifica el nombre de la vía de acceso completa del archivo (en el servidor) que contendrá mensajes sobre la operación de importación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si el parámetro tiene un valor nulo, no se creará ningún archivo para los mensajes de DB2 Spatial Extender.

Los mensajes que se graban en el archivo de mensajes pueden ser:

- Mensajes informativos, como, por ejemplo, un resumen de la operación de importación

- Mensajes de error para datos que no se han podido importar, por ejemplo, debido a sistemas distintos de coordenadas

Estos mensajes corresponden a los datos que se almacenan en el archivo de excepciones (identificado por el parámetro *archivo_excepción*).

El procedimiento almacenado, que se ejecuta como un proceso que es propiedad del propietario de la instancia de DB2, debe tener los privilegios necesarios para crear el archivo. Si el archivo ya existe, el procedimiento almacenado devolverá un error.

El tipo de datos de este parámetro es VARCHAR(256).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Notas de uso:

El procedimiento almacenado ST_import_shape utiliza de uno a cuatro a archivos:

- El archivo principal de formas (extensión .shp). Este archivo es necesario.
- El archivo de índices de formas (extensión .shx). Este archivo es opcional. Si está presente, es posible que mejore el rendimiento de la operación de importación.
- Un archivo dBASE que contiene datos de atributos (extensión .dbf). Este archivo es necesario sólo si se van a importar datos de atributos.
- El archivo de proyección que especifica el sistema de coordenadas de los datos de formas (extensión .prj). Este archivo es opcional. Si este archivo está presente, el sistema de coordenadas que está definido en el mismo se compara con el sistema de coordenadas del sistema de referencia espacial que especifica el parámetro *id_srs*.

La tabla siguiente describe cómo los tipos de datos de atributos dBASE se correlacionan con tipos de datos de DB2. Los demás tipos de datos de atributos no están soportados.

Tabla 30. Relación entre tipos de datos de DB2 y tipos de datos de atributos de dBASE

Tipo .dbf	Longitud .dbfb (Ver nota)	Decimales .dbfb (Ver nota)	Tipo SQL	Comentarios
N	< 5	0	SMALLINT	

Tabla 30. Relación entre tipos de datos de DB2 y tipos de datos de atributos de dBASE (continuación)

Tipo .dbf	Longitud .dbfb (Ver nota)	Decimales .dbfb (Ver nota)	Tipo SQL	Comentarios
N	< 10	0	INTEGER	
N	< 20	0	BIGINT	
N	*len*	*dec*	DECIMAL(*len*,*dec*)	*len*<32
F	*len*	*dec*	REAL	*len* + *dec* < 7
F	*len*	*dec*	DOUBLE	
C	*len*		CHAR(*len*)	
L			CHAR(1)	
D			DATE	

Nota: Esta tabla incluye las siguientes variables, las cuales están definidas ambas en la cabecera del archivo dBASE:

- *len*, que representa la longitud total de la columna en el archivo dBASE. DB2 Spatial Extender utiliza este valor con dos propósitos:
  - Para definir la precisión para el tipo de datos DECIMAL de SQL o la longitud para el tipo de datos CHAR de SQL
  - Para determinar cuál de los tipos de número entero o de coma flotante se utilizará
- *dec*, que representa el número máximo de dígitos situados a la derecha de una coma decimal de la columna en el archivo dBASE. DB2 Spatial Extender utiliza este valor para definir la escala para el tipo de datos DECIMAL de SQL.

Por ejemplo, supongamos que el archivo dBASE contiene una columna de datos cuya longitud (*len*) está definida como 20. Supongamos que el número de dígitos situados a la derecha de la coma decimal (*dec*) está definido como 5. Cuando DB2 Spatial Extender importa datos de esta columna, utiliza los valores de *len* y *dec* para obtener el siguiente tipo de datos de SQL: DECIMAL(20,5).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_import_shape. Este ejemplo utiliza una llamada CALL de DB2 para importar un archivo de formas denominado `/tmp/officesShape` a la tabla denominada OFFICES:

```
call db2gse.ST_import_shape('/tmp/officesShape',NULL,'USA_SRS_1',NULL,
                     'OFFICES',NULL,0,NULL,'LOCATION',NULL,NULL,NULL,NULL,
                     NULL,NULL,NULL,NULL,'/tmp/import_msg',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_export_shape" en la página 263

ST_register_geocoder

Utilice este procedimiento almacenado para registrar un geocodificador que no sea el geocodificador DB2SE_USA_GEOCODER, que se proporciona con DB2 Spatial Extender. DB2 Spatial Extender registra el geocodificador DB2SE_USA_GEOCODER cuando se habilita la base de datos.

Requisitos previos: Antes de registrar un geocodificador:

- Asegúrese de que la función que ejecuta el geocodificador ya esté creada. Se puede registrar cada función de geocodificador como un geocodificador con un nombre de geocodificador identificado de forma exclusiva.
- Obtenga información del proveedor del geocodificador, por ejemplo:
  - La sentencia de SQL que crea la función
  - Los valores que se deben utilizar con los parámetros ST_create_srs de forma que se dé soporte a datos geométricos
  - Información para registrar el geocodificador, por ejemplo:
    - Una descripción del geocodificador
    - Descripciones de los parámetros para el geocodificador
    - Los valores por omisión de los parámetros del geocodificador

El tipo de retorno de la función de geocodificador debe coincidir con el tipo de datos de la columna geocodificada. Los parámetros de geocodificación pueden ser un nombre de columna (denominado *columna de geocodificación*) que contiene datos que necesita el geocodificador. Por ejemplo, los parámetros del geocodificador pueden identificar direcciones o un valor de una significación especial para el geocodificador, tal como el grado mínimo de coincidencia. Si el parámetro de geocodificación es un nombre de columna, la columna debe estar en la misma tabla o vista que la columna geocodificada.

El tipo de retorno de la función de geocodificador sirve como el tipo de datos para la columna geocodificada. El tipo de retorno puede ser cualquier tipo de datos de DB2, tipo definido por el usuario o tipo estructurado. Si se devuelve un tipo definido por el usuario o un tipo estructurado, la función de geocodificador es responsable de devolver un valor válido del tipo de datos respectivo. Si la función de geocodificador devuelve valores de un tipo especial, que es ST_Geometry o uno de sus subtipos, la función de geocodificador es responsable de reestructurar un geometría válida. La geometría se debe representar utilizando un sistema de referencia espacial existente. La geometría es válida si al invocar a la función espacial ST_IsValid para la geometría, el resultado es un 1. Los datos devueltos desde la función de geocodificador se actualizan o se insertan en la tabla geocodificada, en función de qué operación (INSERT o UPDATE) haya causado la generación del valor geocodificado.

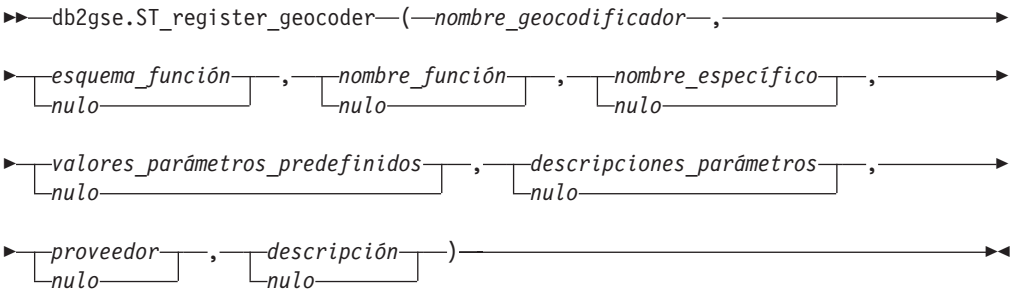
Para saber si un geocodificador ya está registrado, consulte la vista de catálogo DB2GSE.ST_GEOCODERS.

Este procedimiento almacenado sustituye a db2gse.gse_register_gc.

Autorización:

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos donde reside el geocodificador que registra este procedimiento almacenado.

Sintaxis:



Descripciones de parámetros:

nombre_geocodificador
Identifica de forma exclusiva el geocodificador. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

esquema_función
Especifica el esquema para la función que implementa este geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, este valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la función.

El valor *esquema_función* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_función
Especifica el nombre no calificado de la función que implementa este geocodificador. La función ya debe estar creada y listada en SYSCAT.ROUTINES.

Para este parámetro, puede especificar un valor nulo si se ha especificado el parámetro *nombre_específico*. Si no se ha especificado el parámetro *nombre_específico*, el valor de *nombre_función*, junto con el valor implícita o explícitamente definido de *esquema_función*, debe identificar de forma exclusiva la función. Si no se ha especificado el parámetro *nombre_función*, DB2 Spatial Extender recupera el valor de *nombre_función* de la vista de catálogo SYSCAT.ROUTINES.

El valor de *nombre_función* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_específico
Identifica el nombre específico de la función que implementa el geocodificador. La función ya debe estar creada y listada en SYSCAT.ROUTINES.

Para este parámetro, puede especificar un valor nulo si el parámetro *nombre_función* está especificado y la combinación de *esquema_función* y *nombre_función* identifica de forma exclusiva la función de geocodificador. Si el

nombre de la función de geocodificador está sobrecargado, el parámetro *nombre_específico* no puede ser un valor nulo. (Un nombre de función está *sobrecargado* si tiene el mismo nombre, pero no los mismos tipos de datos de parámetro o parámetros, que una o varias de las otras funciones.)

El valor de *nombre_específico* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

valores_parámetros_predefinidos
Especifica la lista de valores por omisión de parámetros de configuración para la función de geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si todo el parámetro *valores_parámetros_predefinidos* tiene un valor nulo, todos los valores por omisión del parámetro tienen un valor nulo.

Si especifica algún valor de parámetro, especifíquelos en el orden en que la función los ha definido y sepárelos mediante una coma. Por ejemplo:

```
valor_parm1_por_omisión,valor_parm2_por_omisión,...
```

Cada valor de parámetro es una expresión de SQL. Siga estas directrices:

- Si un valor es una cadena de caracteres, póngalo entre comillas dobles.
- Si un valor de parámetro es un número, no lo ponga entre comillas simples.
- Si el valor del parámetro es un valor nulo, asígnelo al tipo correcto. Por ejemplo, en lugar de especificar simplemente NULL, especifique:
  ```
  CAST(NULL AS INTEGER)
  ```
- Si el parámetro de geocodificación va a ser una columna de geocodificación, no especifique el valor por omisión del parámetro.

Si algún valor de parámetro no está especificado (es decir, si especifica dos comas consecutivas (`...,,...`)), este parámetro se debe especificar al configurar la geocodificación o al ejecutar la geocodificación en modalidad de proceso por lotes con el parámetro *valores_parámetros* de los procedimientos almacenados respectivos.

El tipo de datos de este parámetro es VARCHAR(32K).

descripciones_parámetros
Especifica la lista de descripciones de parámetros de configuración para la función de geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si todo el parámetro *descripciones_parámetros* tiene un valor nulo, todas las descripciones del parámetro tendrán un valor nulo. Cada descripción de parámetro que especifica explica el significado y la utilización del parámetro, y puede tener una longitud máxima de 256. Las descripciones para los parámetros deben estar separadas por comas y deben aparecer en el orden de los parámetros como están definidos por la función. Si se utilizará una coma dentro de la descripción de un parámetro, ponga la cadena de caracteres entre comillas simples o dobles. Por ejemplo:

```
descripción,'descripción2, que contiene una coma',descripción3
```

El tipo de datos de este parámetro es VARCHAR(32K).

proveedor
Especifica el proveedor que ha implementado el geocodificador. Aunque debe

especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se registra ninguna información sobre el proveedor que ha implementado el geocodificador.

El tipo de datos de este parámetro es VARCHAR(128).

descripción
Describe el geocodificador explicando su aplicación. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro es nulo, no se graba ninguna información de descripción sobre el geocodificador.

Recomendación: Incluya la información siguiente:

- Nombre del sistema de coordenadas si se devolverán datos espaciales, como por ejemplo, texto convencional (WKT) o binario convencional (WKB)
- Sistema de referencia espacial, si se devolverán ST_Geometry o alguno de sus subtipos
- Nombre del área geográfica a la que se aplica este geocodificador
- Cualquier otra información sobre el geocodificador que los usuarios deban saber

El tipo de datos de este parámetro es VARCHAR(256).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo presume que el usuario desea crear un geocodificador que toma la latitud y la longitud como datos de entrada y geocodifica en datos espaciales ST_Point. Para hacer esto, cree primero una función denominada lat_long_gc_func. A continuación, registre un geocodificador denominado SAMPLEGC, que utiliza la función lat_long_gc_func.

Aquí tiene un ejemplo de la sentencia de SQL que crea la función lat_long_gc_func que devuelve ST_Point:

```
CREATE FUNCTION lat_long_gc_func(latitude double,
   longitude double, srId integer)
   RETURNS db2gse.ST_Point
   LANGUAGE SQL
   RETURN db2gse.ST_Point(latitude, longitude, srId)
```

Después de crear la función, puede registrarla como un codificador. Este ejemplo muestra cómo utilizar el mandato CALL del procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_register_geocoder para registrar un geocodificador denominado SAMPLEGC con la función lat_long_gc_func:

```
call db2gse.ST_register_geocoder ('SAMPLEGC',NULL,'LAT_LONG_GC_FUNC',',,1'
            ,NULL,'My Company','Latitude/Longitude to
            ST_Point Geocoder'?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_unregister_geocoder" en la página 290

ST_register_spatial_column

Utilice este procedimiento almacenado para registrar una columna espacial y para asociar un sistema de referencia espacial (SRS) al mismo. Cuando se procesa este procedimiento almacenado, la información sobre la columna espacial que se está registrando se añade a la vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS. Si se registra una columna espacial, se crea una restricción en la tabla, si es posible, para asegurar que todas las geometrías utilicen el SRS especificado.

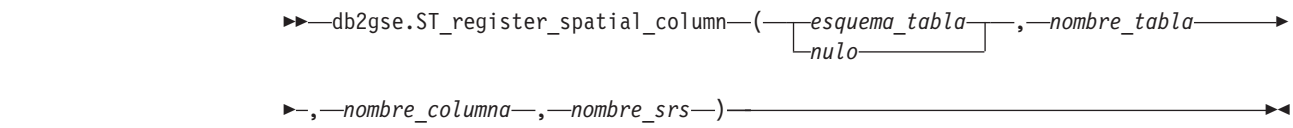
Este procedimiento almacenado sustituye a db2gse.gse_register_layer.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla a la que pertenece la columna espacial que se está registrando
- Privilegio CONTROL o ALTER sobre esta tabla

Sintaxis:



Descripciones de parámetros:

esquema_tabla
Especifica el esquema al que pertenece la tabla o vista que está especificada en el parámetro *nombre_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_tabla
Especifica el nombre no calificado de la tabla o vista que contiene la columna que se está registrando. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_columna
Especifica la columna que se está registrando. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_srs
Especifica el sistema de referencia espacial que se utiliza para esta columna espacial. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_srs* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_register_spatial_column. Este ejemplo utiliza un mandato CALL de DB2 para registrar la columna espacial denominada LOCATION en la tabla denominada CUSTOMERS. Este mandato CALL especifica el valor del parámetro *nombre_srs* como USA_SRS_1:

```
call db2gse.ST_register_spatial_column(NULL,'CUSTOMERS','LOCATION',
     'USA_SRS_1',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

• "ST_unregister_spatial_column" en la página 291

ST_remove_geocoding_setup

Utilice este procedimiento almacenado para eliminar toda la información de configuración de geocodificación para la columna geocodificada.

Este procedimiento almacenado elimina de las vistas de catálogo DB2GSE.ST_GEOCODING y DB2GSE.ST_GEOCODING_PARAMETERS información que está asociada con la columna geocodificada especifica.

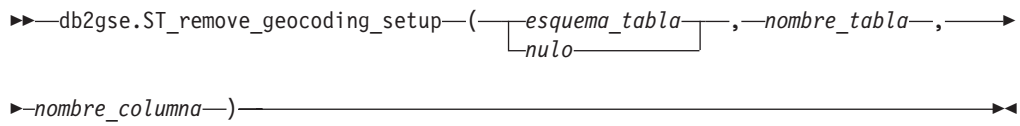
Restricción: no puede eliminar una configuración de geocodificación si la geocodificación está habilitada para la columna geocodificada.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla sobre las que operará el geocodificador especificado
- Privilegio CONTROL o UPDATE sobre la tabla

Sintaxis:



Descripciones de parámetros:

esquema_tabla
Especifica el esquema al que pertenece la tabla o vista que está especificada en el parámetro *nombre_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_tabla
Especifica el nombre no calificado de la tabla o vista que contiene la columna en la que se van a insertar los datos geocodificados o en la que se van a actualizar los mismos. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_columna
Especifica la columna en la que se van a insertar los datos geocodificados o se van a actualizar los mismos. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_columna* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos para este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_remove_geocoding_setup. Este ejemplo utiliza un mandato CALL de DB2 para eliminar la configuración de geocodificación para la tabla denominada CUSTOMER y la columna denominada LOCATION:

```
call db2gse.ST_remove_geocoding_setup(NULL, 'CUSTOMERS', 'LOCATION',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_setup_geocoding" en la página 286

ST_run_geocoding

Utilice este procedimiento almacenado para ejecutar un geocodificador en modalidad de proceso por lotes en una columna geocodificada.

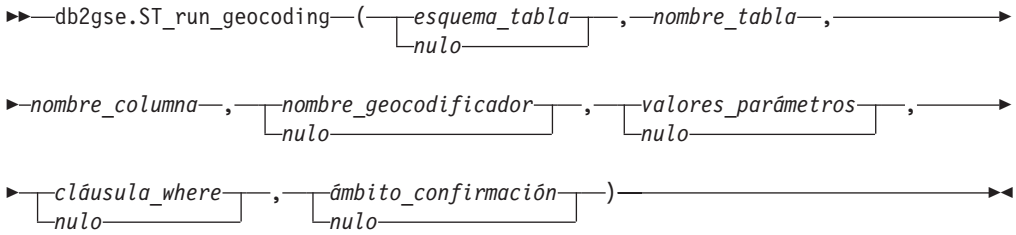
Este procedimiento almacenado sustituye a db2gse.gse_run_gc.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla sobre las que operará el geocodificador especificado
- Privilegio CONTROL o UPDATE sobre la tabla

Sintaxis:



Descripciones de parámetros:

esquema_tabla
Especifica el esquema al que pertenece la tabla o vista que está especificada en el parámetro *nombre_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_tabla
Especifica el nombre no calificado de la tabla o vista que contiene la columna en la que se van a insertar los datos geocodificados o en la que se van a actualizar los mismos. Si hay especificado un nombre de vista, la vista debe ser una vista actualizable. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_columna
Especifica la columna en la que se van a insertar los datos geocodificados o se van a actualizar los mismos. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_columna* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_geocodificador
Especifica el geocodificador que va a realizar la geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, la geocodificación la realiza el geocodificador que se ha especificado al configurar la geocodificación.

El valor de *nombre_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

valores_parámetros
Especifica la lista de valores de parámetros de configuración para la función de geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si todo el parámetro *valores_parámetros* tiene un valor nulo, los valores que se utilizan son los valores de los parámetros que se han especificado al configurar el geocodificador o los valores de los parámetros por omisión para el geocodificador si el geocodificador no se ha configurado.

Si especifica algún valor de parámetro, especifíquelos en el orden en que la función los ha definido y sepárelos mediante una coma. Por ejemplo:

```
parámetro1-valor,parámetro2-valor,...
```

Cada valor de parámetro puede ser un nombre de columna, una cadena de caracteres, un valor numérico o un valor nulo.

Cada valor de parámetro es una expresión de SQL. Siga estas directrices:

- Si un valor de parámetro es un nombre de columna de geocodificación, asegúrese de que la columna esté en la misma tabla o vista donde está ubicada la columna codificada.
- Si un valor de parámetro es una cadena de caracteres, póngalo entre comillas simples.
- Si un valor de parámetro es un número, no lo ponga entre comillas simples.
- Si el parámetro tiene un valor nulo, asígnelo al tipo correcto. Por ejemplo, en lugar de especificar simplemente NULL, especifique:

  ```
  CAST(NULL AS INTEGER)
  ```

Si algún valor de parámetro no está especificado (es decir, si especifica dos comas consecutivas (`...,,...`)), este parámetro se debe especificar al configurar la geocodificación o al ejecutar la geocodificación en modalidad de proceso por lotes con el parámetro *valores_parámetros* de los procedimientos almacenados respectivos.

El tipo de datos de este parámetro es VARCHAR(32K).

cláusula_where
Especifica el cuerpo de la cláusula WHERE, donde define una restricción sobre el conjunto de registros que se van a geocodificar. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si el parámetro *cláusula_where* tiene un valor nulo, el comportamiento resultante depende de si se ha configurado la geocodificación para la columna

(especificada en el parámetro *nombre_columna*) antes de ejecutar el procedimiento almacenado. Si el parámetro *cláusula_where* tiene un valor nulo y:

- Se ha especificado un valor al configurar la geocodificación, dicho valor se utilizará para el parámetro *cláusula_where*.
- No se ha configurado la geocodificación o no se ha especificado ningún valor al configurar la geocodificación, no se utilizará ninguna cláusula where.

Puede especificar una cláusula que se refiera a cualquier columna de la tabla o vista sobre la que vaya a operar el geocodificador. No especifique la palabra clave WHERE.

El tipo de datos de este parámetro es VARCHAR(32K).

ámbito_confirmación
Especifica que se a va realizar un COMMIT después de geocodificar *n* registros. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo.

Si el parámetro *ámbito_confirmación* tiene un valor nulo, el comportamiento resultante depende de si se ha configurado la geocodificación para la columna (especificada en el parámetro *nombre_columna*) antes de ejecutar el procedimiento almacenado. Si el parámetro *ámbito_confirmación* tiene un valor nulo y:

- Se ha especificado un valor al configurar la geocodificación para la columna, dicho valor se utilizará para el parámetro *ámbito_confirmación*.
- No se ha configurado la geocodificación o se ha configurado pero no se ha especificado ningún valor, se utilizará el valor por omisión de 0 (cero) y no se realizará ningún COMMIT.

El tipo de datos de este parámetro es INTEGER.

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_run_geocoding. Este ejemplo utiliza un mandato CALL de DB2 para geocodificar la columna LOCATION en la tabla denominada CUSTOMER. Este mandato CALL especifica el valor del parámetro *nombre_geocodificador* como DB2SE_USA_GEOCODER y el valor del parámetro *ámbito_confirmación* como 10. Se va a realizar un COMMIT después de cada 10 registros geocodificados:

```
call db2gse.ST_run_geocoding(NULL, 'CUSTOMERS', 'LOCATION',
     'DB2SE_USA_GEOCODER',NULL,NULL,10,?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_setup_geocoding" en la página 286

ST_setup_geocoding

Utilice este procedimiento almacenado para asociar una columna que se va a geocodificar con un geocodificador y para configurar los parámetros de geocodificación correspondientes. La información que se configura aquí se registra en las vistas de catálogo DB2GSE.ST_GEOCODING y DB2GSE.ST_GEOCODING_PARAMETERS.

Este procedimiento almacenado no invoca a la geocodificación. Proporciona una manera para que el usuario especifique los valores de los parámetros para la columna que se va a codificar. Con estos valores, se puede realizar con una interfaz mucho más sencilla la invocación subsiguiente de la geocodificación de proceso por lotes o la geocodificación automática. Los valores de los parámetros que se especifican en este paso de configuración alteran temporalmente cualquiera de los valores por omisión de los parámetros para el geocodificador que se han especificado cuando el geocodificador se ha registrado. También puede alterar temporalmente estos valores de parámetros ejecutando el procedimiento almacenado ST_run_geocoding en modalidad de proceso por lotes.

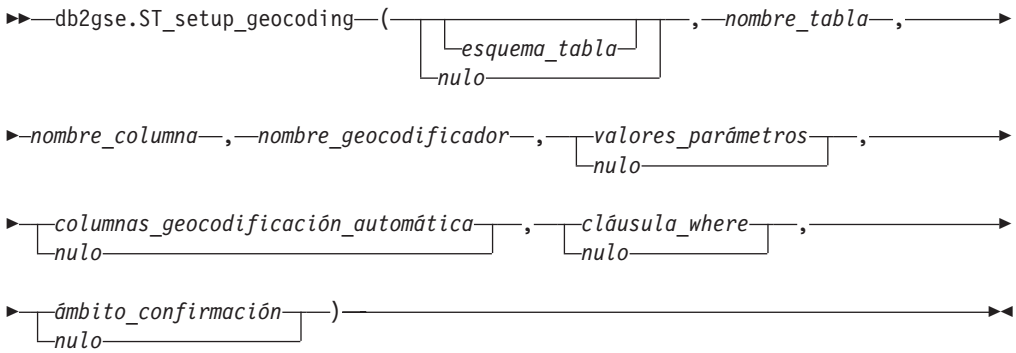
Este paso es un requisito necesario para la geocodificación automática. No puede habilitar la geocodificación automática sin configurar primero los parámetros de geocodificación. Este paso es un requisito necesario para la geocodificación de proceso por lotes. Puede ejecutar la geocodificación en modalidad de proceso por lotes con o sin realizar este paso de configuración. Sin embargo, si el paso de configuración se realiza antes de la geocodificación de proceso por lotes, los valores de los parámetros se toman del momento de configuración si no se han especificado en el momento de la ejecución.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla sobre las que operará el geocodificador especificado
- Privilegio CONTROL o UPDATE sobre la tabla

Sintaxis:



Descripciones de parámetros:

esquema_tabla
Especifica el esquema al que pertenece la tabla o vista que está especificada en el parámetro *nombre_tabla*. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_tabla
Especifica el nombre no calificado de la tabla o vista que contiene la columna en la que se van a insertar los datos geocodificados o en la que se van a actualizar los mismos. Si hay especificado un nombre de vista, la vista debe ser actualizable. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_columna
Especifica la columna en la que se van a insertar los datos geocodificados o se van a actualizar los mismos. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_columna* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_geocodificador
Especifica el geocodificador que va a realizar la geocodificador. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

valores_parámetros
Especifica la lista de valores de parámetros de configuración para la función de

geocodificador. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si todo el parámetro *valores_parámetros* tiene un valor nulo, los valores que se utilizan se toman de los valores por omisión que tenían los parámetros cuando se registró el geocodificador.

Si especifica valores de parámetros, especifíquelos en el orden en que la función los ha definido y sepárelos mediante una coma. Por ejemplo:

```
parámetro1-valor,parámetro2-valor,...
```

Cada valor de parámetro es una expresión de SQL y puede ser un nombre de columna, una cadena de caracteres, un valor numérico o un valor nulo. Siga estas directrices:

- Si un valor de parámetro es un nombre de columna de geocodificación, asegúrese de que la columna esté en la misma tabla o vista donde está ubicada la columna codificada.
- Si un valor de parámetro es una cadena de caracteres, póngalo entre comillas simples.
- Si un valor de parámetro es un número, no lo ponga entre comillas simples.
- Si el valor del parámetro está especificado como un valor nulo, asígnelo al tipo correcto. Por ejemplo, en lugar de especificar simplemente NULL, especifique:

  ```
  CAST(NULL AS INTEGER)
  ```

Si algún valor de parámetro no está especificado (es decir, si especifica dos comas consecutivas (`...,,...`)), este parámetro se debe especificar al configurar la geocodificación o al ejecutar la geocodificación en modalidad de proceso por lotes con el parámetro *valores_parámetros* de los procedimientos almacenados respectivos.

El tipo de datos de este parámetro es VARCHAR(32K).

columnas_geocodificación_automática
Especifica la lista de nombres de columna en la que se creará el desencadenante. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo y la geocodificación está habilitada, una actualización de cualquier columna de la tabla hace que se active el desencadenante.

Si especifica un valor para el parámetro *columnas_geocodificación_automática*, especifique nombres de columna en cualquier orden y separe los nombres de columna con una coma. El nombre de columna debe existir en la misma tabla donde se ubica la columna geocodificada.

El valor de este parámetro se aplica solamente a la geocodificación subsiguiente.

El tipo de datos de este parámetro es VARCHAR(32K).

cláusula_where
Especifica el cuerpo de la cláusula WHERE, donde define una restricción sobre el conjunto de registros que se van a geocodificar. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, no se define ninguna restricción en la cláusula WHERE.

La cláusula puede hacer referencia a cualquier columna de la tabla o vista sobre la que va a operar el geocodificador. No especifique la palabra clave WHERE.

El valor de este parámetro se aplica solamente a la geocodificación en modalidad de proceso por lotes subsiguiente.

El tipo de datos de este parámetro es VARCHAR(32K).

ámbito_confirmación
Especifica que se va a realizar un COMMIT después de codificar *n* registros. Aunque debe especificar un valor para este parámetro, el valor puede ser nulo. Si este parámetro tiene un valor nulo, se realiza un COMMIT después de que se geocodifiquen todos los registros.

El valor de este parámetro se aplica solamente a la geocodificación en modalidad de proceso por lotes subsiguiente.

El tipo de datos de este parámetro es INTEGER.

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_setup_geocoding. Este ejemplo utiliza un mandato CALL de DB2 para configurar un proceso de geocodificación para la columna geocodificada denominada LOCATION en la tabla denominada CUSTOMER. Este mandato CALL especifica el valor del parámetro *nombre_geocodificador* como DB2SE_USA_GEOCODER:

```
call db2gse.ST_setup_geocoding(NULL, 'CUSTOMERS', 'LOCATION',
    'DB2SE_USA_GEOCODER','ADDRESS,CITY,STATE,ZIP,1,100,80,,,,"$HOME/sqllib/
     gse/refdata/ky.edg","$HOME/sqllib/samples/spatial/EDGESample.loc"',
     'ADDRESS,CITY,STATE,ZIP',NULL,10,?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- “ST_unregister_geocoder” en la página 290
- “ST_remove_geocoding_setup” en la página 281

ST_unregister_geocoder

Utilice este procedimiento almacenado para deshacer el registro de un geocodificador distinto de DB2SE_USA_GEOCODER, que se proporciona con DB2 Spatial Extender.

Restricción: No puede deshacer el registro de un geocodificador si está especificado en la configuración de geocodificación para cualquier columna.

Para determinar si un geocodificador está especificado en la configuración de geocodificación para una columna, comprueba las vista de catálogo DB2GSE.ST_GEOCODING y DB2GSE.ST_GEOCODING_PARAMETERS. Para buscar información sobre el geocodificador del que desea deshacer el registro, consulte la vista de catálogo DB2GSE.ST_GEOCODERS.

Este procedimiento almacenado sustituye a db2gse.gse_unregist_gc.

Autorización:

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos donde reside el geocodificador que se debe desregistrar.

Sintaxis:

Descripciones de parámetros:

nombre_geocodificador
Identifica de forma exclusiva el geocodificador. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_geocodificador* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_unregister_geocoder. Este ejemplo utiliza un mandato CALL de DB2 para deshacer el registro del geocodificador denominado SAMPLEGC:

```
call db2gse.ST_unregister_geocoder('SAMPLEGC',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_register_geocoder" en la página 275
- "ST_setup_geocoding" en la página 286

ST_unregister_spatial_column

Utilice este procedimiento almacenado para eliminar el registro de una columna espacial. El procedimiento almacenado elimina el registro haciendo lo siguiente:

- Eliminando la asociación del sistema de referencia espacial con la columna espacial. La vista de catálogo ST_GEOMETRY_COLUMNS sigue conteniendo la columna espacial, pero la columna ya no está asociada con ningún sistema de referencia espacial.
- Para una tabla base, descartando la restricción que DB2 Spatial Extender ha puesto sobre esta tabla para asegurar que los valores de la geometría en esta columna espacial estén todos representados en el mismo sistema de referencia espacial

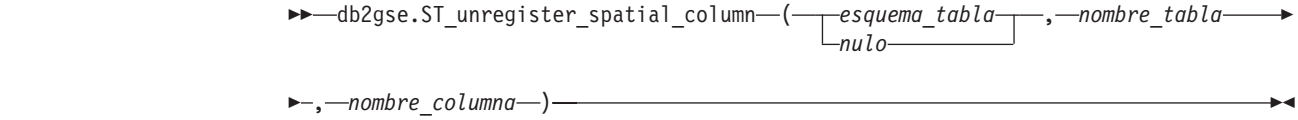
Este procedimiento almacenado sustituye a db2gse.gse_unregist_layer.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio CONTROL o ALTER sobre esta tabla

Sintaxis:



Descripciones de parámetros:

esquema_tabla
Especifica el esquema al que pertenece la tabla que está especificada en el parámetro *nombre_tabla*. Aunque debe especificar un valor para este parámetro,

el valor puede ser nulo. Si este parámetro tiene un valor nulo, el valor en el registro especial CURRENT SCHEMA se utiliza como el nombre de esquema para la tabla o vista.

El valor de *esquema_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_tabla
Especifica el nombre no calificado de la tabla que contiene la columna que está especifica en el parámetro *nombre_columna*. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_tabla* se convierte a mayúsculas a menos que esté entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

nombre_columna
Especifica la columna espacial de la que desea deshacer el registro. Debe especificar un valor que no sea nulo para este parámetro.

El valor de *nombre_columna* se convierte a mayúsculas a menos que lo ponga entre comillas dobles.

El tipo de datos de este parámetro es VARCHAR(128) o, si pone el valor entre comillas dobles, VARCHAR(130).

Parámetros de salida:

msg_code
Especifica el código de mensaje que se devuelve del procedimiento almacenado. El valor de este parámetro de salida identifica la condición de error, de finalización o de aviso que se ha encontrado durante el proceso del procedimiento. Si el valor de este parámetro es para una condición de finalización satisfactoria o de aviso, el procedimiento ha finalizado su tarea. Si el valor del parámetro es para una condición de error, no se han realizado cambios en la base de datos.

El tipo de datos de este parámetro de salida es INTEGER.

msg_text
Especifica el texto del mensaje propiamente dicho, correspondiente al código de mensaje, que se devuelve desde el procedimiento almacenado. El texto del mensaje puede incluir información adicional sobre la condición de finalización satisfactoria, aviso o error, como, por ejemplo, dónde se ha encontrado un error.

El tipo de datos de este parámetro de salida es VARCHAR(1024).

Ejemplo:

Este ejemplo muestra cómo utilizar el procesador de línea de mandatos de DB2 para invocar al procedimiento almacenado ST_unregister_spatial_column. Este ejemplo utiliza un mandato CALL de DB2 para deshacer el registro de la columna espacial denominada LOCATION en la tabla denomina CUSTOMERS:

```
call db2gse.ST_unregister_spatial_column(NULL,'CUSTOMERS','LOCATION',?,?)
```

Los dos signos de interrogación al final de este mandato CALL representan los parámetros de salida, *msg_code* y *msg_text*. Los valores de estos parámetros de salida se visualizan después de que se ejecute el procedimiento almacenado.

Información relacionada:

- "ST_register_spatial_column" en la página 279

Capítulo 21. Vistas de catálogo

Las vistas de catálogo de Spatial Extender contienen información acerca de:

"Vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS"
Sistemas de coordenadas que puede utilizar

"Vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS" en la página 296
Columnas espaciales que puede llenar o actualizar

"Vista de catálogo DB2GSE.ST_GEOCODERS" en la página 299 y "Vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS" en la página 301
Geocodificadores que puede utilizar

"Vista de catálogo DB2GSE.ST_GEOCODING" en la página 299 y "Vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS" en la página 301
Especificaciones para configurar un geocodificador para que se ejecute automáticamente y para configurar, anticipadamente, operaciones que se deben realizar durante la geocodificación de proceso por lotes.

"Vista de catálogo DB2GSE.ST_SIZINGS" en la página 302
Longitudes máximas permitidas que puede asignar a las variables.

"Vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS" en la página 303 Sistemas de referencia espacial que puede utilizar.

"Vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE" en la página 306
Las unidades de medida (metros, millas, pies, etc.) en las que se pueden generar las funciones espaciales.

Vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS

Consulte la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS para recuperar información sobre sistemas de coordenadas registrados. Spatial Extender registra automáticamente los sistemas de coordenadas en el catálogo de Spatial en las siguientes ocasiones:

- Cuando habilita una base de datos para operaciones espaciales.
- Cuando los usuarios definen sistemas de coordenadas adicionales en la base de datos.

Para ver una descripción de las columnas de esta vista, consulte la tabla siguiente.

Tabla 31. Columnas de la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS

Nombre	Tipo de datos	¿Anulable?	Contenido
COORDSYS_NAME	VARCHAR(128)	No	Nombre de este sistema de coordenadas. El nombre es exclusivo dentro de la base de datos.

Tabla 31. Columnas de la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
COORDSYS_TYPE	VARCHAR(128)	No	Tipo de este sistema de coordenadas: **PROJECTED** Bidimensional. **GEOGRAPHIC** Tridimensional. Utiliza coordenadas X e Y. **GEOCENTRIC** Tridimensional. Utiliza coordenadas X, Y y Z. **UNSPECIFIED** Sistema de coordenadas abstracto o que no es del mundo real. El valor de esta columna se obtiene de la columna DEFINITION.
DEFINITION	VARCHAR(2048)	No	Representación de WKT (texto convencional) de la definición de este sistema de coordenadas.
ORGANIZATION	VARCHAR(128)	Sí	Nombre de la organización (por ejemplo, un cuerpo de estándares como el ESPG (European Petrol Survey Group)) que define este sistema de coordenadas. Esta columna tiene un valor nulo si la columna ORGANIZATION_COORDSYS_ID tiene un valor nulo.
ORGANIZATION_ COORDSYS_ID	INTEGER	Sí	Identificador numérico asignado a este sistema de coordenadas por la organización que ha definido el sistema de coordenadas. Este identificador y el valor de la columna ORGANIZATION identifican de forma exclusiva el sistema de coordenadas a menos que el identificador y el valor tengan ambos un valor nulo. Si la columna ORGANIZATION tiene un valor nulo, la columna ORGANIZATION_COORDSYS_ID también tiene un valor nulo.
DESCRIPTION	VARCHAR(256)	Sí	Descripción del sistema de coordenadas que indica su aplicación.

Vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS

Utilice la vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS para buscar información sobre todas las columnas espaciales en todas las tablas que contienen datos espaciales en la base de datos. Si se ha registrado una columna espacial en asociación con un sistema de referencia espacial, también puede utilizar la vista para saber el nombre y el identificador numérico del sistema de referencia espacial. Para obtener información adicional sobre las columnas espaciales, consulte la vista de catálogo SYSCAT.COLUMN de DB2.

Para ver una descripción de DB2GSE.ST_GEOMETRY_COLUMNS, consulte la tabla siguiente.

Tabla 32. Columnas de la vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS

Nombre	Tipo de datos	¿Anulable?	Contenido
TABLE_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece la tabla que contiene esta columna espacial.
TABLE_NAME	VARCHAR(128)	No	Nombre no calificado de la tabla que contiene esta columna espacial.
COLUMN_NAME	VARCHAR(128)	No	Nombre de esta columna espacial. La combinación de TABLE_SCHEMA, TABLE_NAME y COLUMN_NAME identifica exclusivamente la columna.
TYPE_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece el tipo de datos declarado de esta columna espacial. Este nombre se obtiene del catálogo de DB2.
TYPE_NAME	VARCHAR(128)	No	Nombre no calificado del tipo de datos declarado de esta columna espacial. Este nombre se obtiene del catálogo de DB2.
SRS_NAME	VARCHAR(128)	Sí	Nombre del sistema de referencia espacial que está asociado con esta columna espacial. Si no hay ningún sistema de referencia espacial asociado a esta columna, SRS_NAME tiene un valor nulo.
SRS_ID	INTEGER	Sí	Identificador numérico del sistema de referencia espacial que está asociado a esta columna espacial. Si no hay ningún sistema de referencia espacial asociado a esta columna, SRS_ID tiene un valor nulo.

Vista de catálogo DB2GSE.ST_GEOCODER_ PARAMETERS

Cuando habilita una base de datos para operaciones espaciales, la información sobre los parámetros del geocodificador proporcionado, DB2GSE_USA_GEOCODER, se registra automáticamente en el catálogo de DB2 Spatial Extender. Si registra geocodificadores adicionales, la información sobre los parámetros de los mismos también se registra en el catálogo. Para recuperar del catálogo la información sobre los parámetros de un geocodificador, consulte la vista de catálogo DB2GSE.ST_GEOCODER_PARAMETERS. Para ver una descripción de las columnas de esta vista, consulte la tabla siguiente.

Para obtener más información sobre los parámetros de los geocodificadores, consulte la vista de catálogo SYSCAT.ROUTINEPARMS de DB2. Para ver una descripción de esta vista, consulte el manual *Consulta de SQL*.

Tabla 33. Columnas de DB2GSE.ST_GEOCODER_PARAMETERS

Nombre	Tipo de datos	¿Anulable?	Contenido
GEOCODER_NAME	VARCHAR(128)	No	Nombre del geocodificador al que pertenece este parámetro.

Tabla 33. Columnas de DB2GSE.ST_GEOCODER_PARAMETERS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
ORDINAL	SMALLINT	No	Posición de este parámetro (es decir, el parámetro especificado en la columna PARAMETER_NAME) en la signatura de la función que sirve como el geocodificador especificado en la columna GEOCODER_NAME. Los valores combinados en las columnas GEOCODER_NAME y ORDINAL identifican de forma exclusiva este parámetro. Un registro en la vista de catálogo SYSCAT.ROUTINEPARMS de DB2 también contiene información sobre este parámetro. Este registro contiene un valor que aparece en la columna ORDINAL de SYSCAT.ROUTINEPARMS. Este valor es el mismo que el que aparece en la columna ORDINAL de la vista DB2GSE.ST_GEOCODER_PARAMETERS.
PARAMETER_NAME	VARCHAR(128)	Sí	Nombre de este parámetro. Si no se ha especificado un nombre cuando se ha creado la función a la que pertenece este parámetro, la columna PARAMETER_NAME tiene un valor nulo. El contenido de la columna PARAMETER_NAME se obtiene del catálogo de DB2.
TYPE_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece este parámetro. Este nombre se obtiene del catálogo de DB2.
TYPE_NAME	VARCHAR(128)	No	Nombre no calificado del tipo de datos de los valores asignados a este parámetro. Este nombre se obtiene del catálogo de DB2.
PARAMETER_DEFAULT	VARCHAR(2048)	Sí	El valor por omisión que se asignará a este parámetro. DB2 interpretará este valor como una expresión de SQL. Si el valor está entre comillas dobles, se enviará al geocodificador como una cadena de caracteres. En caso contrario, la evaluación de la expresión de SQL determinará qué tipo de datos de parámetros será cuando se envíe al geocodificador. Si la columna PARAMETER_DEFAULT contiene un valor nulo, este valor nulo se enviará al geocodificador. El valor por omisión puede tener un valor correspondiente en la vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS. También puede tener un valor correspondiente en los datos de entrada del procedimiento almacenado ST_run_geocoding. Si cualquiera de los valores correspondientes difiere del valor por omisión, el valor correspondiente alterará temporalmente el valor por omisión.
DESCRIPTION	VARCHAR(256)	Sí	Descripción del parámetro que indica su aplicación.

Vista de catálogo DB2GSE.ST_GEOCODERS

Cuando habilita una base de datos para operaciones espaciales, el geocodificador proporcionado, DB2GSE_USA_GEOCODER, se registra automáticamente en el catálogo de DB2 Spatial Extender. Cuando desee hacer que geocodificadores adicionales estén disponibles para los usuarios, necesita registrar estos geocodificadores. Para recuperar información sobre los geocodificadores registrados, consulte la vista de catálogo DB2GSE.ST_GEOCODERS. Para ver una descripción de las columnas de esta vista, consulte la tabla siguiente.

Para ver información sobre los parámetros de los geocodificadores, consulte la vista de catálogo DB2GSE.ST_GEOCODER_PARAMETERS de DB2 Spatial Extender y la vista de catálogo SYSCAT.ROUTINEPARMS de DB2. Para obtener información sobre las funciones que se utilizan como geocodificadores, consulte la vista de catálogo SYSCAT.ROUTINES de DB2.

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

Nombre	Tipo de datos	¿Anulable?	Contenido
GEOCODER_NAME	VARCHAR(128)	No	Nombre de este geocodificador. Es exclusivo en la base de datos.
FUNCTION_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece la función que está siendo utilizado como este geocodificador.
FUNCTION_NAME	VARCHAR(128)	No	Nombre no calificado de la función que está siendo utilizada como este geocodificador.
SPECIFIC_NAME	VARCHAR(128)	No	Nombre específico de la función que está siendo utilizado como este geocodificador. Los valores combinados de FUNCTION_SCHEMA y SPECIFIC_NAME identifican de forma exclusiva la función que está siendo utilizada como este geocodificador.
RETURN_TYPE_SCHEMA	VARCHAR(128)	No	Nombre del esquema al que pertenece el tipo de datos del parámetro de salida de este geocodificador. Este nombre se obtiene del catálogo de DB2.
RETURN_TYPE_NAME	VARCHAR(128)	No	Nombre no calificado del tipo de datos del parámetro de salida de este geocodificador. Este nombre se obtiene del catálogo de DB2.
VENDOR	VARCHAR(256)	Sí	Nombre del proveedor que ha creado este geocodificador.
DESCRIPTION	VARCHAR(256)	Sí	Descripción del geocodificador que indica su aplicación.

Vista de catálogo DB2GSE.ST_GEOCODING

Cuando configura las operaciones de geocodificación, los detalles de la configuración se registran automáticamente en el catálogo de DB2 Spatial Extender. Para conocer estos detalles, consulte las vistas de catálogo DB2GSE.ST_GEOCODING y DB2GSE.ST_GEOCODING_PARAMETERS. La vista de catálogo DB2GSE.ST_GEOCODING, que se describe en la tabla siguiente, contiene detalles de todas las configuraciones; por ejemplo, el número de registros que un geocodificador procesará antes de cada confirmación. La vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS contiene detalles que son específicos

para de geocodificador. Por ejemplo, las configuraciones para el geocodificador proporcionado, DB2GSE_USA_GEOCODER, incluyen el grado mínimo en el que deben coincidir las direcciones proporcionadas como datos de entrada y las direcciones reales para que el geocodificador geocodifique los datos de entrada. Este requisito mínimo, denominado *grado mínimo de coincidencia* se registra en la vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS.

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

Nombre	Tipo de datos	¿Anulable?	Contenido
TABLE_SCHEMA	VARCHAR(128)	No	Nombre del esquema que contiene la tabla que contiene la columna identificada en la columna COLUMN_NAME.
TABLE_NAME	VARCHAR(128)	No	Nombre no calificado de la tabla que contiene la columna identificada en la columna COLUMN_NAME.
COLUMN_NAME	VARCHAR(128)	No	Nombre de la columna espacial que se llenará según las especificaciones que se muestran en esta vista de catálogo. Los valores combinados de las columnas TABLE_SCHEMA, TABLE_NAME y COLUMN_NAME identifican de forma exclusiva la columna espacial.
GEOCODER_NAME	VARCHAR(128)	No	Nombre del geocodificador que producirá datos para la columna espacial especificada en la columna COLUMN_NAME. Sólo se puede asignar un geocodificador a una columna espacial.
MODE	VARCHAR(128)	No	Modalidad para el proceso de geocodificación: **BATCH** Sólo está habilitada la geocodificación de proceso por lotes. **AUTO** La geocodificación automática está configurada y activada. **INVALID** Se ha detectado una inconsistencia en las tablas de catálogos espaciales; la entrada de geocodificación no es válida.
SOURCE_COLUMNS	VARCHAR(10000)	Sí	Nombres de las columnas de tabla configuradas para geocodificación automática. Siempre que se actualizan estas columnas, un desencadenante indica al geocodificador que geocodifique los datos actualizados.
WHERE_CLAUSE	VARCHAR(10000)	Sí	Condición de búsqueda dentro de una cláusula WHERE. Esta condición indica que cuando un geocodificador se ejecuta en modalidad de proceso por lotes, sólo geocodifica datos dentro del subconjunto especificado de registros.
COMMIT_COUNT	INTEGER	Sí	Número de filas que se procesarán durante la geocodificación de proceso por lotes antes de la emisión de una confirmación. Si el valor de la columna COMMIT_COUNT es 0 (cero) o nulo, no se emite ninguna confirmación.

Vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS

Cuando configura las operaciones de geocodificación para un geocodificador determinado, los aspectos de la configuración específicos del geocodificador se registran automáticamente en el catálogo de Spatial Extender. Por ejemplo, una operación específica a un geocodificador determinado, DB2GSE_USA_GEOCODER, es comparar las direcciones proporcionadas como datos de entrada con unos datos de referencia y geocodificar las anteriores si coinciden con los últimos en un cierto grado, o en un grado superior al especificado. Cuando configura operaciones para este geocodificador, especifica cuál será este grado, denominado *grado mínimo de coincidencia*; y esta especificación se registra en el catálogo.

Para conocer los aspectos de configuración específicos del codificador para las operaciones de geocodificación, consulte la vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS. Esta vista se describe en la tabla siguiente.

Algunos valores por omisión para la configuración de las operaciones de geocodificación están disponibles en la vista de catálogo DB2GSE.ST_GEOCODER_PARAMETERS. Los valores en la vista DB2GSE.ST_GEOCODING_PARAMETERS alteran temporalmente los valores por omisión.

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

Nombre	Tipo de datos	¿Anulable?	Contenido
TABLE_SCHEMA	VARCHAR(128)	No	Nombre del esquema que contiene la tabla que contiene la columna identificada en la columna COLUMN_NAME.
TABLE_NAME	VARCHAR(128)	No	Nombre no calificado de la tabla que contiene la columna espacial.
COLUMN_NAME	VARCHAR(128)	No	Nombre de la columna espacial que se llenará según las especificaciones que se muestran en esta vista de catálogo. Los valores combinados en las columnas TABLE_SCHEMA, TABLE_NAME y COLUMN_NAME identifican de forma exclusiva esta columna espacial.
ORDINAL	SMALLINT	No	Posición de este parámetro (es decir, el parámetro especificado en la columna PARAMETER_NAME) en la signatura de la función que sirve como geocodificador para la columna identificada en la columna COLUMN_NAME. Un registro en la vista de catálogo SYSCAT.ROUTINEPARMS de DB2 también contiene información sobre este parámetro. Este registro contiene un valor que aparece en la columna ORDINAL de SYSCAT.ROUTINEPARMS. Este valor es el mismo que el que aparece en la columna ORDINAL de la vista DB2GSE.ST_GEOCODING_PARAMETERS.

Tabla 36. Columnas de la vista de catálogo DB2GSE.ST_GEOCODING_PARAMETERS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
PARAMETER_NAME	VARCHAR(128)	Sí	Nombre de un parámetro en la definición del geocodificador. Si no se ha especificado ningún nombre al definir el geocodificador, PARAMETER_NAME tiene un valor nulo. Este contenido de la columna PARAMETER_NAME se obtiene del catálogo de DB2.
PARAMETER_VALUE	VARCHAR(2048)	Sí	El valor que se asigna a este parámetro. DB2 interpretará este valor como una expresión de SQL. Si el valor está entre comillas dobles, se enviará al geocodificador como una cadena de caracteres. En caso contrario, la evaluación de la expresión de SQL determinará qué tipo de datos de parámetros será cuando se envíe al geocodificador. Si la columna PARAMETER_VALUE contiene un valor nulo, este valor nulo se envía al geocodificador. La columna PARAMETER_VALUE corresponde a la columna PARAMETER_DEFAULT en la vista de catálogo DB2GSE.ST_GEOCODER_PARAMETERS. Si la columna PARAMETER_VALUE contiene un valor, este valor altera temporalmente el valor por omisión en la columna PARAMETER_DEFAULT. Si la columna PARAMETER_VALUE tiene un valor nulo, se utilizará el valor por omisión.

Vista de catálogo DB2GSE.ST_SIZINGS

Utilice la vista de catálogo DB2GSE.ST_SIZINGS para recuperar:

- Todas las variables soportadas por Spatial Extender; por ejemplo, *nombre de sistema de coordenadas*, *nombre de geocodificador* y variables a las que se puede asignar representaciones WKT de datos espaciales.
- La longitud máxima permitida, si se conoce, de valores asignados a estas variables (por ejemplo, las longitudes máximas permitidas de nombres de sistemas de coordenadas, de nombres de geocodificadores y de representaciones WKT de datos espaciales).

Para ver una descripción de las columnas en la vista, consulte la tabla siguiente.

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

Nombre	Tipo de datos	¿Anulable?	Contenido
VARIABLE_NAME	VARCHAR(128)	No	Término que indica una variable. El término es exclusivo dentro de la base de datos.

Tabla 37. Columnas de la vista de catálogo DB2GSE.ST_SIZINGS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
SUPPORTED_VALUE	INTEGER	Sí	Longitud máxima permitida de los valores asignados a la variable mostrada en la columna VARIABLE_NAME. Los valores posibles de la columna SUPPORTED_VALUE son: **Un valor numérico distinto de 0** La longitud máxima permitida de valores asignados a esta variable. **0** Se permite cualquier longitud o no se puede determinar la longitud permitida. **NULL** Spatial Extender no da soporte a esta variable.
DESCRIPTION	VARCHAR(128)	Sí	Descripción de esta variable.

Vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS

Consulte la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS para recuperar información sobre los sistemas de referencia espacial registrados. Spatial Extender registra automáticamente los sistemas de referencia espacial en el catálogo de Spatial Extender en las siguientes ocasiones:

- Cuando habilita una base de datos para operaciones espaciales, cinco sistemas de referencia especial geodésicos por omisión y 318 sistemas de referencia espacial geodésicos predefinidos. Para obtener detalles, consulte los apartados "Cómo decidirse entre la utilización de un sistema de referencia espacial por omisión o la creación de un sistema nuevo" en la página 70 y "Sistemas de referencia soportados por DB2 Geodetic Extender" en la página 220.
- Cuando los usuarios crean sistemas de referencia espacial adicionales.

Para obtener el máximo partido de la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, necesita entender que cada sistema de referencia espacial está asociado a un sistema de coordenadas. El sistema de referencia espacial está diseñado en parte para convertir coordenadas derivadas del sistema de coordenadas en valores que DB2 pueda procesar con la máxima eficacia, y en parte para definir la extensión máxima posible de espacio a las que estas coordenadas pueden hacer referencia.

Para saber el nombre y el tipo de sistema de coordenadas asociado a un sistema de referencia espacial determinado, consulte las columnas COORDSYS_NAME y COORDSYS_TYPE de la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS. Para obtener más información acerca del sistema de coordenadas, consulte la vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Tabla 38. Columnas de la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS

Nombre	Tipo de datos	¿Anulable?	Contenido
SRS_NAME	VARCHAR(128)	No	Nombre del sistema de referencia espacial. Este nombre es exclusivo dentro de la base de datos.

Tabla 38. Columnas de la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
SRS_ID	INTEGER	No	Identificador numérico del sistema de referencia espacial. Cada sistema de referencia espacial tiene un identificador numérico exclusivo. Los sistemas de referencia espacial geodésicos tienen valores SRS_ID comprendidos entre 2000000000 y 2000001000. Las funciones espaciales especifican sistemas de referencia espacial por sus identificadores numéricos en lugar de hacerlo por sus nombres.
X_OFFSET	DOUBLE	No	El desplazamiento se puede restar a todas las coordenadas X de una geometría. La sustracción es un paso en el proceso de convertir las coordenadas de la geometría en valores que DB2 pueda procesar con la máxima eficacia. Un paso subsiguiente es multiplicar el número resultante de la resta por el factor de escala que se muestra en la columna X_SCALE.
X_SCALE	DOUBLE	No	El factor de escala por el que se multiplica el número resultante de la operación de restar un desplazamiento a una coordenada X. Este factor es idéntico al valor que se muestra en la columna Y_SCALE.
Y_OFFSET	DOUBLE	No	El desplazamiento que se debe restar a todas las coordenadas Y de una geometría. La sustracción es un paso en el proceso de convertir las coordenadas de la geometría en valores que DB2 pueda procesar con la máxima eficacia. Un paso subsiguiente es multiplicar el número resultante de la sustracción por el factor de escala que se muestra en la columna Y_SCALE.
Y_SCALE	DOUBLE	No	Factor de escala por el que se multiplica el número resultante de la operación de restar un desplazamiento a una coordenada Y. Este factor es idéntico al valor que se muestra en la columna X_SCALE.
Z_OFFSET	DOUBLE	No	Desplazamiento que se debe restar a todas las coordenadas Z de una geometría. La sustracción es un paso en el proceso de convertir las coordenadas de la geometría en valores que DB2 pueda procesar con la máxima eficacia. Un paso subsiguiente es multiplicar el número resultante de la suscripción por el factor de escala que se muestra en la columna Z_SCALE.
Z_SCALE	DOUBLE	No	Factor de escala por el que se multiplica el número resultante de la operación de restar un desplazamiento a una coordenada Z.
M_OFFSET	DOUBLE	No	Desplazamiento que se debe restar a todas las medidas asociadas a una geometría. La sustracción es un paso en el proceso de convertir las medidas en valores que DB2 pueda procesar con la máxima eficacia. Un paso subsiguiente es multiplicar el número resultante de la sustracción por el factor de escala que se muestra en la columna M_SCALE.

Tabla 38. Columnas de la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
M_SCALE	DOUBLE	No	Factor de escala por el que se multiplica el número resultante de sustraer el desplazamiento a una medida.
MIN_X	DOUBLE	No	Mínimo valor posible para coordenadas X en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores de las columnas X_OFFSET y X_SCALE.
MAX_X	DOUBLE	No	Máximo valor posible para las coordenadas X en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores de las columnas X_OFFSET y X_SCALE.
MIN_Y	DOUBLE	No	Mínimo valor posible para las coordenadas Y en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas Y_OFFSET y Y_SCALE.
MAX_Y	DOUBLE	No	Máximo valor posible para las coordenadas Y en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas Y_OFFSET y Y_SCALE.
MIN_Z	DOUBLE	No	Mínimo valor posible para las coordenadas Z en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas Z_OFFSET y Z_SCALE.
MAX_Z	DOUBLE	No	Máximo valor posible para las coordenadas Z en las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas Z_OFFSET y Z_SCALE.
MIN_M	DOUBLE	No	Mínimo valor posible para las medidas que se pueden almacenar con geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas M_OFFSET y M_SCALE.
MAX_M	DOUBLE	No	Máximo valor posible para las medidas que se pueden almacenar con las geometrías a las que se aplica este sistema de referencia espacial. Este valor se obtiene a partir de los valores en las columnas M_OFFSET y M_SCALE.
COORDSYS_NAME	VARCHAR(128)	No	Nombre identificador del sistema de coordenadas en las que se basa este sistema de referencia espacial.
COORDSYS_TYPE	VARCHAR(128)	No	Tipo de sistema de coordenadas en el que se basa este sistema de referencia espacial.
ORGANIZATION	VARCHAR(128)	Sí	Nombre de la organización (por ejemplo, un cuerpo de estándares) que ha definido el sistema de coordenadas en el que se basa este sistema de referencia espacial. ORGANIZATION es nulo si ORGANIZATION_COORSYS_ID es nulo.

Tabla 38. Columnas de la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS (continuación)

Nombre	Tipo de datos	¿Anulable?	Contenido
ORGANIZATION_ COORDSYS_ID	INTEGER	Sí	Nombre de la organización (por ejemplo, un cuerpo de estándares) que ha definido el sistema de coordenadas en el que se basa este sistema de referencia espacial. ORGANIZATION_COORDSYS_ID es nulo si ORGANIZATION es nulo.
DEFINITION	VARCHAR(2048)	No	Representación de WKT (texto convencional) de la definición de este sistema de coordenadas.
DESCRIPTION	VARCHAR(256)	Sí	Descripción del sistema de referencia espacial.

Conceptos relacionados:

- "Sistemas de referencia espacial" en la página 68

Tareas relacionadas:

- "Creación de un sistema de referencia espacial" en la página 76

Vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE

Algunas funciones espaciales aceptan o devuelven valores que indican una distancia determinada. En algunos casos, puede seleccionar en que unidad de medida se expresará la distancia. Por ejemplo, ST_Distance devuelve la distancia mínima entre dos geometrías específicas. En alguna ocasión es posible que requiera que ST_Distance devuelva la distancia en términos de millas; en otra ocasión, es posible que requiera una distancia expresada en términos de metros. Para saber las unidades de medida entre las que puede seleccionar, consulte la vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE.

Tabla 39. Columnas de la vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE

Nombre	Tipo de datos	¿Anulable?	Contenido
UNIT_NAME	VARCHAR(128)	No	Nombre de la unidad de medida. Este nombre es exclusivo en la base de datos.
UNIT_TYPE	VARCHAR(128)	No	Tipo de la unidad de medida. Los valores posibles son: **LINEAR** La unidad de medida es lineal. **ANGULAR** La unidad de medida es angular.
CONVERSION_FACTOR	DOUBLE	No	Valor numérico utilizado para convertir esta unidad de medida en su unidad base. La unidad base para las unidades lineales de medida es METER; la unidad base para unidades angulares de medida es RADIAN. La unidad base propiamente dicha tiene un factor de conversión de 1.0.
DESCRIPTION	VARCHAR(256)	Sí	Descripción de la unidad de medida.

Capítulo 22. Funciones espaciales: categorías y usos

Este capítulo proporciona información básica sobre todas las funciones espaciales y las organiza en categorías.

Funciones espaciales

DB2® Spatial Extender proporciona funciones que:

- Convierten geometrías entre varios formatos de intercambio de datos. Estas funciones se denominan *funciones constructoras*.
- Comparan geometrías para obtener información sobre perímetros, intersecciones y otros elementos. Estas funciones se denominan *funciones de comparación*.
- Devuelven información sobre las propiedades de las geometrías como, por ejemplo, las coordenadas y medidas de las geometrías, las relaciones entre geometrías y los perímetros.
- Generan nuevas geometrías a partir de geometrías existentes.
- Miden la distancia más corta entre puntos de geometrías.
- Proporcionan información sobre parámetros del índice.
- Proporcionan proyecciones y conversiones entre distintos sistemas de coordenadas.

Conceptos relacionados:

- "Función que proporciona información sobre las distancias" en la página 344
- "Función que proporciona información sobre el índice" en la página 344
- "Conversiones entre sistemas de coordenadas" en la página 345

Información relacionada:

- "Ejemplos del funcionamiento de las funciones espaciales" en la página 127
- "Funciones que proporcionan información sobre las propiedades de las geometrías" en la página 329
- "Funciones que comparan elementos geográficos" en la página 316
- "Funciones que generan nuevas geometrías a partir de geometrías existentes" en la página 336
- "Funciones espaciales que convierten valores de geometría en formatos de intercambio de datos" en la página 307

Funciones espaciales que convierten valores de geometría en formatos de intercambio de datos

DB2 Spatial Extender proporciona funciones espaciales que convierten geometrías entre los siguientes formatos de intercambio de datos:

- Representación de texto convencional (WKT)
- Representación binaria convencional (WKB)
- Representación de forma ESRI
- Representación GML (Geography Markup Language)

Las funciones para crear geometrías a partir de estos formatos se denominan *funciones constructoras*.

Conceptos relacionados:

- "Visión general de las funciones constructoras" en la página 308
- "Conversión a una representación de texto convencional (WKT)" en la página 312
- "Conversión a una representación binaria convencional (WKB)" en la página 313
- "Conversión a la representación de forma ESRI" en la página 314
- "Conversión a la representación GML (Geography Markup Language)" en la página 315

Información relacionada:

- "Grupos de transformación" en la página 523

Visión general de las funciones constructoras

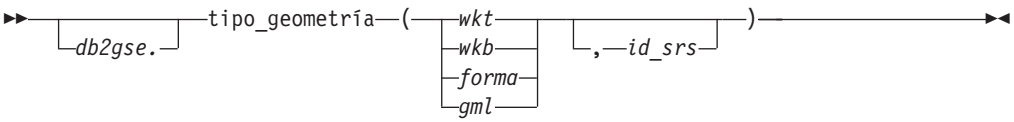
Las funciones constructoras tienen el mismo nombre que el tipo de datos de datos de geometría de la columna en la que se insertarán los datos. Estas funciones operan de forma coherente sobre cada uno de los formatos de intercambio de datos proporcionados como entrada. Esta sección proporciona:

- El SQL para invocar a funciones que operan sobre formatos de intercambio de datos, y el tipo de geometría devuelto por estas funciones
- El SQL para invocar a una función que crea puntos a partir de coordenadas X e Y y el tipo de geometría devuelto por esta función
- Ejemplos de código y conjuntos resultantes

Funciones que operan sobre formatos de intercambio de datos

Esta sección proporciona la sintaxis para invocar a funciones que operan sobre formatos de intercambio de datos, describe los parámetros de entrada de las funciones e identifica el tipo de geometría devuelto por estas funciones.

Sintaxis:



Parámetros y otros elementos de la sintaxis:

db2gse Nombre del esquema al que pertenecen los tipos de datos espaciales proporcionados por DB2® Spatial Extender.

tipo_geometría
Puede ser una de las funciones constructoras siguientes:

- ST_Point
- ST_LineString
- ST_Polygon
- ST_MultiPoint
- ST_MultiLineString

- ST_MultiPolygon
- ST_GeomCollection
- ST_Geometry

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la geometría.

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la geometría.

forma Un valor de tipo BLOB(2G) que contiene la representación de forma ESRI de la geometría.

gml Valor de tipo CLOB(2G) que contiene la representación GML (Geography Markup Language) de la geometría.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la geometría resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Tipo de retorno:

tipo_geometría

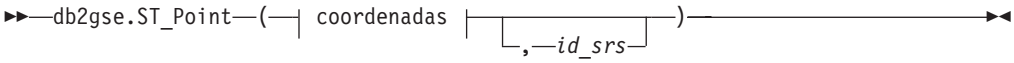
Si *tipo_geometría* es *ST_Geometry*, el tipo dinámico del tipo de geometría devuelto corresponde a la geometría indicada por el valor de entrada.

Si *tipo_geometría* es cualquier otro tipo, el tipo dinámico del tipo de geometría devuelto corresponde al nombre de la función. Si la geometría indicada por el valor de entrada no coincide con el nombre de función ni con el nombre de uno de sus subtipos, se emite un error.

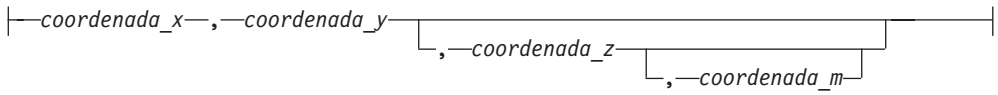
Función que crea geometrías a partir de coordenadas

La función ST_Point crea geometrías no solo a partir de formatos de intercambio de datos, sino también a partir de valores de coordenadas numéricos. Esto es muy útil si los datos de ubicación ya están almacenados en la base de datos. Esta sección proporciona la sintaxis para invocar a la función ST_Point, una descripción de sus parámetros, e información sobre el tipo de geometría devuelto por la función.

Sintaxis:



coordenadas:



Parámetros:

coordenada_x
Valor de tipo DOUBLE que especifica la coordenada X del punto resultante.

coordenada_y
Valor de tipo DOUBLE que especifica la coordenada Y del punto resultante.

coordenada_z
Valor de tipo DOUBLE que especifica la coordenada Z del punto resultante.

Si se omite el parámetro *coordenada_z*, el punto resultante no tendrá coordenada Z. Para ese punto, el resultado de ST_Is3D será un 0 (cero).

coordenada_m
Valor de tipo DOUBLE que especifica la coordenada M del punto resultante.

Si se omite el parámetro *coordenada_m*, el punto resultante no tendrá una medida. Para ese punto, el resultado de ST_IsMeasured será un 0 (cero).

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al punto resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_Point

Ejemplos

Esta sección proporciona ejemplos de código para invocar a funciones constructoras, código para crear tablas donde guardar los datos resultantes de las funciones constructoras, código para recuperar los datos resultantes y los propios datos resultantes.

El ejemplo siguiente inserta una fila en la tabla SAMPLE_GEOMETRY con un ID igual a 100 y un valor de punto con una coordenada X igual a 30, una coordenada Y igual a 40, dentro del sistema de referencia espacial 1 utilizando la representación de coordenadas y la representación de texto convencional (WKT). Luego inserta otra fila con un ID igual a 200 y un valor de cadena lineal con las coordenadas indicadas.

```
CREATE TABLE sample_geometry (id INT, geom db2gse.ST_Geometry);

INSERT INTO sample_geometry(id, geom)
       VALUES(100,db2gse.ST_Geometry('point(30 40)', 1));

INSERT INTO sample_geometry(id, geom)
       VALUES(200,db2gse.ST_Geometry('linestring(50 50, 100 100', 1));

SELECT id, TYPE_NAME(geom) FROM sample_geometry

ID      2
------  --------------------------------
   100 "ST_POINT"
   200  "ST_LINESTRING"
```

Si sabemos que la columna espacial sólo puede contener valores de ST_Point, podemos utilizar el ejemplo siguiente para insertar dos puntos. Si se intenta insertar una cadena lineal o cualquier otro tipo distinto de un punto, se producirá un error de SQL. La primera inserción crea una geometría de punto a partir de la representación de texto convencional (WKT). La segunda inserción crea una geometría de punto a partir de valores de coordenadas numéricos. Observe que estos valores de entrada también se podrían seleccionar a partir de columnas de tabla existentes.

```
CREATE TABLE sample_points (id INT, geom db2gse.ST_Point);

INSERT INTO sample_points(id, geom)
       VALUES(100,db2gse.ST_Point('point(30 40)', 1));

INSERT INTO sample_points(id, geom)
       VALUES(101,db2gse.ST_Point(50, 50, 1));

SELECT id, TYPE_NAME(geom) FROM sample_geometry

ID      2
------  ------------------------------
   100 "ST_POINT"
   101 "ST_POINT"
```

El ejemplo siguiente utiliza SQL intercalado y supone que la aplicación llena las áreas de datos con los valores apropiados.

```
              EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS CLOB(10000) wkt_buffer;
   SQL TYPE IS CLOB(10000) gml_buffer;
   SQL TYPE IS BLOB(10000) wkb_buffer;
   SQL TYPE IS BLOB(10000) shape_buffer;
 EXEC SQL END DECLARE SECTION;

// * Insertar aquí el código de aplicación para insertar datos */

EXEC SQL INSERT INTO sample_geometry(id, geom)
         VALUES(:id, db2gse.ST_Geometry(:wkt_buffer,1));

EXEC SQL INSERT INTO sample_geometry(id, geom)
         VALUES:id, db2gse.ST_Geometry(:wkb_buffer,1));

EXEC SQL INSERT INTO sample_geometry(id, geom)
         VALUES(:id, db2gse.ST_Geometry(:gml_buffer,1));

EXEC SQL INSERT INTO sample_geometry(id, geom)
         VALUES(:id, db2gse.ST_Geometry(:shape_buffer,1));
```

El siguiente código Java™ de ejemplo utiliza JDBC para insertar geometrías de punto utilizando valores de coordenadas numéricos X, Y, y la representación WKT para especificar las geometrías.

```
String ins1 = "INSERT into sample_geometry (id, geom)
              VALUES(?, db2gse.ST_PointFromText(CAST( ?
              as VARCHAR(128)), 1))";
PreparedStatement pstmt = con.prepareStatement(ins1);
pstmt.setInt(1, 100);                    // id value
pstmt.setString(2, "point(32.4 50.7)");  // wkt value
int rc = pstmt.executeUpdate();

String ins2 = "INSERT into sample_geometry (id, geom)
              VALUES(?, db2gse.ST_Point(CAST( ? as double),
              CAST(? as double), 1))";
pstmt = con.prepareStatement(ins2);
```

```
pstmt.setInt(1, 200);         // id value
pstmt.setDouble(2, 40.3);     // lat
pstmt.setDouble(3, -72.5);    // long
rc = pstmt.executeUpdate();
```

Información relacionada:

• "Grupos de transformación" en la página 523

Conversión a una representación de texto convencional (WKT)

Las representaciones de texto son valores CLOB que representan cadenas de caracteres ASCII. Permiten el intercambio de geometrías en forma de texto ASCII.

La función **ST_AsText** convierte en una cadena de caracteres WKT un valor de geometría contenido en una tabla. El ejemplo siguiente realiza una simple consulta desde la línea de mandatos para seleccionar valores que previamente se insertaron en la tabla SAMPLE_GEOMETRY.

```
SELECT id, VARCHAR(db2gse.ST_AsText(geom), 50) AS WKTGEOM
   FROM sample_geometry;
ID   WKTGEOM
------  --------------------------------
   100   POINT ( 30.00000000 40.00000000)
   200   LINESTRING ( 50.00000000 50.00000000, 100.00000000 100.00000000)
```

El ejemplo siguiente utiliza SQL intercalado para seleccionar los valores que previamente se insertaron en la tabla SAMPLE_GEOMETRY.

```
               EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS CLOB(10000) wkt_buffer;
   short wkt_buffer_ind = -1;
 EXEC SQL END DECLARE SECTION;

 EXEC SQL
 SELECT id, db2gse.ST_AsText(geom)
  INTO :id, :wkt_buffer :wkt_buffer_ind
   FROM sample_geometry
  WHERE id = 100;
```

Como alternativa, se puede utilizar el grupo de transformación ST_WellKnownText para convertir implícitamente geometrías en su representación de texto convencional al realizar su vinculación. El código de ejemplo siguiente muestra cómo utilizar el grupo de transformación.

```
              EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS CLOB(10000) wkt_buffer;
   short wkt_buffer_ind = -1;
 EXEC SQL END DECLARE SECTION;

 EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownText;

 EXEC SQL
   SELECT id, geom
  INTO :id, :wkt_buffer :wkt_buffer_ind
   FROM sample_geometry
  WHERE id = 100;
```

La sentencia SELECT no utiliza ninguna función espacial para convertir la geometría.

Además de las funciones tratadas en esta sección, DB2® Spatial Extender proporciona otras funciones que también convierten geometrías a partir de representaciones de texto convencionales. DB2 Spatial Extender proporciona estas otras funciones para cumplir la especificación "Simple Features for SQL" de OGC y el estándar "SQL/MM Part 3: Spatial" de ISO. Estas funciones son:

- **ST_WKTToSQL**
- **ST_GeomFromText**
- **ST_GeomCollFromTxt**
- **ST_PointFromText**
- **ST_LineFromText**
- **ST_PolyFromText**
- **ST_MPointFromText**
- **ST_MLineFromText**
- **ST_MPolyFromText**

Información relacionada:

• "Grupos de transformación" en la página 523

Conversión a una representación binaria convencional (WKB)

La representación WKB consta de estructuras de datos binarios que deben ser valores BLOB. Estos valores BLOB representan estructuras de datos binarios que deben ser gestionadas por un programa de aplicación escrito en un lenguaje de programación soportado por DB2® y para el cual DB2 tiene una vinculación de lenguaje.

La función **ST_AsBinary** convierte un valor de geometría contenido en una tabla en la representación binaria convencional (WKB) que se puede insertar en una variable BLOB de la memoria del programa. El ejemplo siguiente utiliza SQL intercalado para seleccionar los valores que previamente se insertaron en la tabla SAMPLE_GEOMETRY.

```
             EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS BLOB(10000) wkb_buffer;
   short wkb_buffer_ind = -1;
 EXEC SQL END DECLARE SECTION;

 EXEC SQL
   SELECT id, db2gse.ST_AsBinary(geom)
   INTO :id, :wkb_buffer :wkb_buffer_ind
   FROM sample_geometry
   WHERE id = 200;
```

Como alternativa, se puede utilizar el grupo de transformación ST_WellKnownBinary para convertir implícitamente geometrías en su representación binaria convencional al realizar su vinculación. El código de ejemplo siguiente muestra cómo utilizar este grupo de transformación.

```
             EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS BLOB(10000) wkb_buffer;
   short wkb_buffer_ind = -1;
 EXEC SQL END DECLARE SECTION;

 EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownBinary;
```

```
EXEC SQL
  SELECT id, geom
  INTO :id, :wkb_buffer :wkb_buffer_ind
  FROM sample_geometry
  WHERE id = 200;
```

La sentencia SELECT no utiliza ninguna función espacial para convertir la geometría.

Además de las funciones tratadas en esta sección, existen otras funciones que también convierten geometrías a partir de representaciones binarias convencionales. DB2 Spatial Extender proporciona estas otras funciones para cumplir la especificación "Simple Features for SQL" de OCG y el estándar "SQL/MM Part 3: Spatial" de ISO. Estas funciones son:

- **ST_WKBToSQL**
- **ST_GeomFromWKB**
- **ST_GeomCollFromWKB**
- **ST_PointFromWKB**
- **ST_LineFromWKB**
- **ST_PolyFromWKB**
- **ST_MPointFromWKB**
- **ST_MLineFromWKB**
- **ST_MPolyFromWKB**

Información relacionada:

- "Grupos de transformación" en la página 523

Conversión a la representación de forma ESRI

La representación de forma ESRI consiste de estructuras de datos binarios que deben ser gestionadas por un programa de aplicación escrito en un lenguaje soportado.

La función **ST_AsShape** convierte un valor de geometría contenido en una tabla en la representación de forma ESRI que se puede insertar en una variable BLOB de la memoria del programa. El ejemplo siguiente utiliza SQL intercalado para seleccionar los valores que previamente se insertaron en la tabla SAMPLE_GEOMETRY.

```
            EXEC SQL BEGIN DECLARE SECTION;
  sqlint32 id;
  SQL TYPE IS BLOB(10000) shape_buffer;
EXEC SQL END DECLARE SECTION;

EXEC SQL
  SELECT id, db2gse.ST_AsShape(geom)
  INTO :id, :shape_buffer
  FROM sample_geometry;
```

Como alternativa, se puede utilizar el grupo de transformación ST_Shape para convertir implícitamente geometrías en su representación de forma al realizar su vinculación. El código de ejemplo siguiente muestra cómo utilizar el grupo de transformación.

```
             EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS BLOB(10000) shape_buffer;
   short shape_buffer_ind = -1;
 EXEC SQL END DECLARE SECTION;

 EXEC SQL
SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

 EXEC SQL
   SELECT id, geom
   FROM sample_geometry
   WHERE id = 300;
```

La sentencia SELECT no utiliza ninguna función espacial para convertir la geometría.

Información relacionada:

- "Grupos de transformación" en la página 523

Conversión a la representación GML (Geography Markup Language)

Las representaciones GML (Geography Markup Language) son cadenas de caracteres ASCII. Permiten el intercambio de geometrías en forma de texto ASCII.

La función **ST_AsGML** convierte en una cadena de texto GML un valor de geometría contenido en una tabla. El ejemplo siguiente selecciona los valores que previamente se insertaron en la tabla SAMPLE_GEOMETRY. Los resultados que aparecen en el ejemplo siguiente se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido variará según la pantalla que se utilice.

```
SELECT id, VARCHAR(db2gse.ST_AsGML(geom), 500) AS GMLGEOM
   FROM sample_geometry;
ID           GMLGEOM
-----------  ------------------------------------------------------------
        100  <gml:Point srsName="EPSG:4269">
             <gml:coord><gml:X>30</gml:X><gml:Y>40</gml:Y></gml:coord>
             </gml:Point>
        200  <gml:LineString srsName="EPSG:4269">
             <gml:coord><gml:X>50</gml:X><gml:Y>50</gml:Y></gml:coord>
             <gml:coord><gml:X>100</gml:X><gml:Y>100</gml:Y></gml:coord>
             </gml:LineString>
```

Como alternativa, se puede utilizar el grupo de transformación ST_GML para convertir implícitamente geometrías en su representación HTML al realizar su vinculación.

```
SET CURRENT DEFAULT TRANSFORM GROUP = ST_GML

SELECT id, geom AS GMLGEOM
   FROM sample_geometry;

ID           GMLGEOM
-----------  ------------------------------------------------------------
        100  <gml:Point srsName="EPSG:4269">
             <gml:coord><gml:X>30</gml:X><gml:Y>40</gml:Y></gml:coord>
             </gml:Point>
        200  <gml:LineString srsName="EPSG:4269">
             <gml:coord><gml:X>50</gml:X><gml:Y>50</gml:Y></gml:coord>
             <gml:coord><gml:X>100</gml:X><gml:Y>100</gml:Y></gml:coord>
             </gml:LineString>
```

La sentencia SELECT no utiliza ninguna función espacial para convertir la geometría.

Información relacionada:

- "Grupos de transformación" en la página 523

Funciones que comparan elementos geográficos

Determinadas funciones espaciales devuelven información sobre las maneras en que los elementos geográficos están relacionados entre sí o son equiparables el uno con el otro. Otras funciones espaciales devuelven información sobre si dos definiciones de sistemas de coordenadas o dos sistemas de referencia espacial son iguales. En todos los casos, la información devuelta es el resultado de comparar entre sí geometrías, definiciones de sistemas de coordenadas o sistemas de referencia espacial. Las funciones que proporcionan esta información se denominan *funciones de comparación.*

Las funciones de comparación son:

- **ST_Contains y ST_Within.** Estas funciones toman dos geometrías como parámetros de entrada y determinan si el interior de una geometría forma intersección con el interior de la otra.
- **ST_Intersects, ST_Crosses, ST_Overlaps y ST_Touches.** Estas funciones devuelven información sobre intersecciones de geometrías.
- **ST_EnvIntersects y ST_MBRIntersects.** Estas funciones determinan si el rectángulo más pequeño que engloba a una geometría forma intersección con el rectángulo más pequeño que engloba a otra geometría.
- **ST_Equals, ST_EqualCoordsys y ST_EqualSRS.** Estas funciones determinan si dos elementos comparados son idénticos.
- **ST_Relate.** Esta función determina si las geometrías comparadas cumplen las condiciones de la serie de la matriz patrón DE-9IM.
- **ST_Disjoint.** Esta función comprueba si no hay intersección entre dos geometrías.

Conceptos relacionados:

- "Función que compara geometrías con la serie de la matriz patrón DE-9IM" en la página 328
- "Visión general de las funciones de comparación" en la página 316
- "Funciones que comprueban si una geometría contiene otra" en la página 318
- "Funciones que comprueban las intersecciones entre geometrías" en la página 321
- "Funciones que comparan envolturas de geometrías" en la página 326
- "Funciones que comprueban si dos elementos son idénticos" en la página 326
- "Función que comprueba si no hay intersección entre dos geometrías" en la página 328

Visión general de las funciones de comparación

Las funciones de comparación de DB2® Spatial Extender devuelven un 1 (uno) si una comparación cumple determinados criterios, el valor 0 (cero) si una comparación no cumple los criterios y un valor nulo si la comparación no se ha podido realizar. No se pueden realizar comparaciones si la operación de comparación no se ha definido para los parámetros de entrada o si cualquiera de

los parámetros es nulo. Las comparaciones *sí* pueden realizarse si a los parámetros se asignan geometrías con tipos de datos o dimensiones diferentes.

El modelo *Dimensionally Extended 9 Intersection Model (DE-9IM)* es un método matemático que define la relación espacial entre geometrías de tipos y dimensiones diferentes. Este modelo expresa relaciones espaciales entre todos los tipos de geometrías en forma de intersecciones por pares de interiores, perímetros y exteriores, teniendo en cuenta la dimensión de las intersecciones resultantes.

Dadas las geometrías *a* y *b*: I(*a*), B(*a*), y E(*a*) representan el interior, perímetro y exterior de *a*, respectivamente. I(*b*), B(*b*), y E(*b*) representan el interior, perímetro y exterior de *b*. Las intersecciones de I(*a*), B(*a*) y E(*a*) con I(*b*), B(*b*) y E(*b*) producen una matriz de 3 por 3. Cada intersección puede dar lugar a geometrías de dimensiones diferentes. Por ejemplo, si la intersección de los perímetros de dos polígonos consta de un punto y una cadena lineal, la función dim devuelve la dimensión máxima: 1.

El resultado de la función dim es un valor igual a –1, 0, 1 ó 2. El valor –1 corresponde al conjunto vacío o dim(null), que se obtiene cuando no se encuentra ninguna intersección.

	Interior	**Perímetro**	**Exterior**
Interior	dim(I(a) ∩ I(b))	dim(I(a) ∩ B(b))	dim(I(a) ∩ E(b))
Perímetro	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 devueltos por las funciones de comparación se pueden interpretar o verificar comparándolos con una matriz patrón que representa los valores aceptables para el modelo DE-9IM.

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

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

F No debe existir una intersección; dim = -1.

***** No importa si existe o no una intersección; dim = -1, 0, 1 ó 2.

0 Debe existir una intersección y su dimensión exacta 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 matriz patrón siguiente correspondiente a la función ST_Within incluye los valores V, F y *.

Tabla 40. Matriz de ST_Within. Matriz patrón de la función ST_Within para combinaciones de geometrías.

	Interior Geometría b	**Perím. Geometría b**	**Exterior Geometría b**
Interior Geometría a	V	*	F
Perím. Geometría a	*	*	F
Exterior Geometría a	*	*	*

La función ST_Within devuelve el valor 1 cuando los interiores de ambas geometrías forman intersección y cuando el interior o perímetro de *a* no forma intersección con el exterior de *b*. Todas las demás condiciones no son significativas.

Cada función tiene como mínimo una matriz patrón, pero algunas necesitan más de una matriz para describir las relaciones de las diversas combinaciones de tipos de geometrías.

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

Lista de funciones

Las funciones de comparación son:

- ST_Contains
- ST_Crosses
- ST_Disjoint
- ST_EnvIntersects
- ST_EqualCoordsys
- ST_Equals
- ST_EqualSRS
- ST_Intersects
- ST_MBRIntersects
- ST_Overlaps
- ST_Relate
- ST_Touches
- ST_Within

Funciones que comprueban si una geometría contiene otra

ST_Contains y ST_Within ambas toman dos geometrías como parámetros de entrada y determinan si el interior de una geometría forma intersección con el interior de la otra. En términos más sencillos, ST_Contains determina si la primera geometría proporcionada a la función engloba a la segunda geometría (es decir, si la primera contiene a la segunda). ST_Within determina si la primera geometría está completamente dentro de la segunda.

ST_Contains

ST_Contains devuelve un 1 (uno) si la segunda geometría está completamente dentro de la primera geometría. La función ST_Contains devuelve un resultado que es exactamente el opuesto al de la función ST_Within.

La Figura 44 en la página 319 muestra ejemplos de ST_Contains:

- Una geometría de multipuntos contiene un punto o geometrías de multipuntos cuando todos los puntos se encuentran dentro de la primera geometría.

- Una geometría de polígono contiene una geometría de multipunto cuando todos los puntos se encuentran en los límites del polígono o en el interior del polígono.
- Una geometría de cadena lineal contiene geometrías de cadena lineal, multipunto o punto cuando todos los puntos se encuentran dentro de la primera geometría.
- Una geometría de polígono contiene geometrías de polígono, cadena lineal o punto cuando la segunda geometría se encuentran en el interior del polígono.

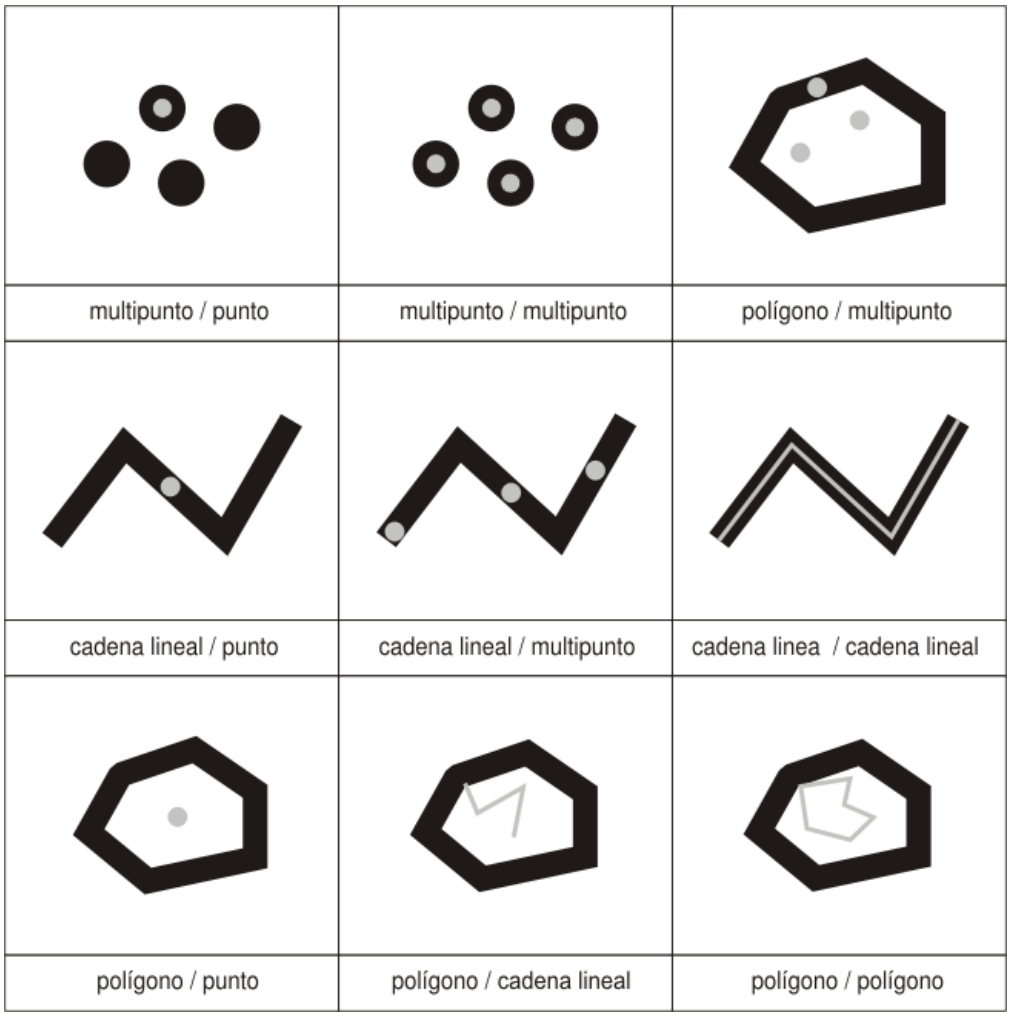


Figura 44. ST_Contains. Las geometrías en negro representan la geometría ″a″ y las geometrías en gris representan la geometría ″b″. En todos los casos, la geometría *a* contiene completamente a la geometría *b*.

La matriz patrón de la función ST_Contains indica que los interiores de ambas geometrías deben formar intersección y que el interior o el perímetro de la secundaria (geometría *b*) no debe formar intersección con el exterior de la primaria (geometría *a*). El asterisco (*) indica que no es significativo el hecho de que exista una intersección entre estas partes de las geometrías.

Tabla 41. Matriz de ST_Contains

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Interior Geometría a	V	*	*
Perím. Geometría a	*	*	*
Exterior Geometría a	F	F	*

ST_Within

ST_Within devuelve un 1 (uno) si la primera geometría está completamente dentro de la segunda geometría. El resultado devuelto por ST_Within es exactamente el opuesto al de ST_Contains.

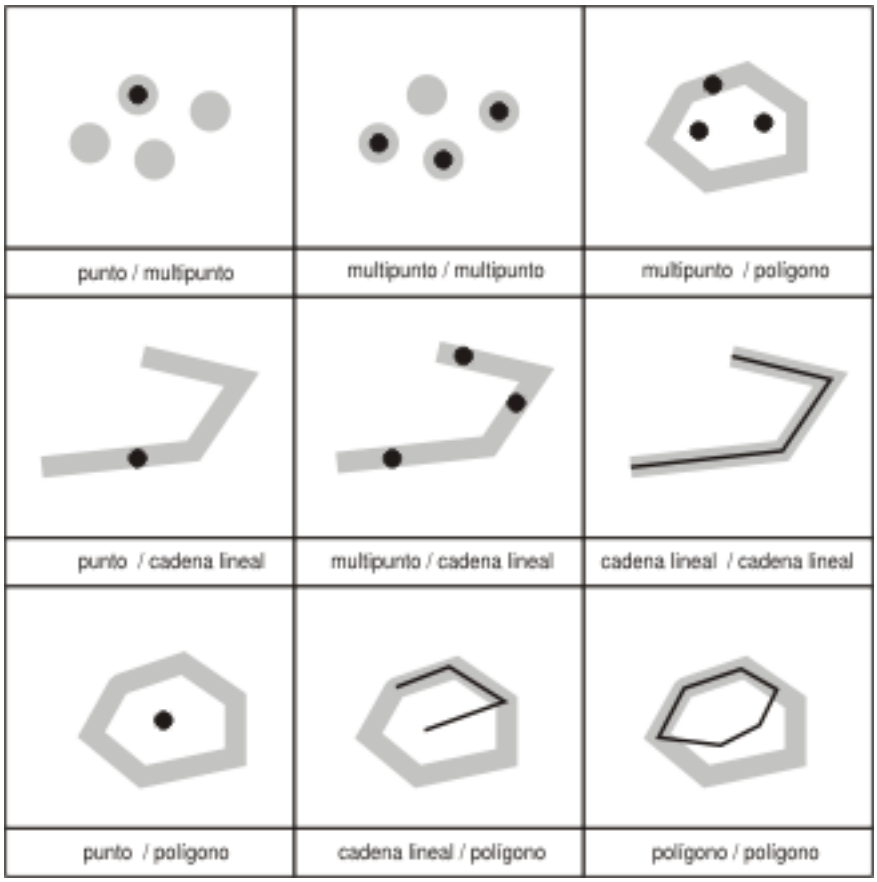


Figura 45. ST_Within

La matriz patrón de la función ST_Within indica que los interiores de ambas geometrías deben formar intersección y que el interior o el perímetro de la primera geometría (geometría *a*) no debe formar intersección con el exterior de la segunda geometría (geometría *b*). El asterisco (*) indica que las intersecciones restantes no son significativas.

Tabla 42. Matriz de ST_Within

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Interior Geometría a	V	*	F
Perím. Geometría a	*	*	F
Exterior Geometría a	*	*	*

La Figura 45 en la página 320 muestra ejemplos de ST_Within:

- Una geometría de punto está dentro de una geometría de multipunto cuando su interior forma intersección con uno de los puntos de la segunda geometría.
- Una geometría de multipunto está dentro de una geometría de multipunto cuando los interiores de todos los puntos forman intersección con la segunda geometría.
- Una geometría de multipunto está dentro de una geometría de polígono cuando todos los puntos se encuentran en los límites del polígono o en el interior del polígono.
- Una geometría de punto está dentro de una geometría de cadena lineal cuando todos los puntos están dentro de la segunda geometría. En la Figura 45 en la página 320, el punto no está dentro de la cadena lineal porque su interior no forma intersección con la cadena lineal; sin embargo, la geometría de multipunto está dentro de la cadena lineal porque todos sus puntos forman intersección con el interior de la cadena lineal.
- Una geometría de cadena lineal está dentro de otras geometrías de cadena lineal cuando todos sus puntos forman intersección con las segunda geometría.
- Una geometría de punto no está dentro de una geometría de polígono porque su interior no forma intersección con el perímetro o el interior del polígono.
- Una geometría de cadena lineal está dentro de una geometría de polígono cuando todos sus puntos forman intersección con el perímetro o el interior del polígono.
- Una geometría de polígono está dentro de una geometría de polígono cuando todos sus puntos forman intersección con el perímetro o el interior del polígono.

Funciones que comprueban las intersecciones entre geometrías

ST_Intersects, ST_Crosses, ST_Overlaps y ST_Touches determinan si una geometría forma intersección con otra. Difieren principalmente respecto al ámbito de intersección que es objeto de comprobación:

- ST_Intersects determina si las dos geometrías proporcionadas cumplen una de estas cuatro condiciones: los interiores de las geometrías forman intersección, sus perímetros forman intersección, el perímetro de la primera geometría forma intersección con el interior de la segunda, o bien que el interior de la primera geometría forma intersección con el perímetro de la segunda.
- ST_Crosses se utiliza para analizar la intersección de geometrías de dimensiones diferentes, con una excepción: también puede analizar la intersección de cadenas lineales. En todos los casos, el propio lugar de intersección se considera que es una geometría; ST_Crosses necesita que la dimensión de esta geometría sea menor que la de las geometrías que forman intersección (o bien, si ambas son cadenas lineales, que la dimensión del lugar de intersección sea menor que una cadena lineal). Por ejemplo, las dimensiones de una cadena lineal y un polígono son 1 y 2, respectivamente. Si estas geometrías forman intersección y el lugar de intersección es lineal (el recorrido de la cadena lineal a lo largo del polígono), entonces ese lugar se puede considerar que es una cadena lineal. Además, debido a que la dimensión de una cadena lineal (1) es menor que la de un polígono (2), ST_Crosses, después de analizar la intersección, devolvería un valor de 1.
- Las geometrías proporcionadas a ST_Overlaps como entrada deben tener la misma dimensión. ST_Overlaps necesita que estas geometrías se solapen parcialmente, formando una nueva geometría (la región de solapamiento) que tenga la misma dimensión que las geometrías originales.

- ST_Touches determina si los perímetros de las dos geometrías forman intersección.

ST_Intersects

ST_Intersects devuelve un 1 (uno) si la intersección de dos geometrías no es un conjunto vacío. El resultado devuelto por ST_Intersects es exactamente el opuesto al de ST_Disjoint.

La función ST_Intersects devuelve un 1 (uno) si la evaluación de las condiciones de cualquiera de las siguientes matrices patrón da un resultado VERDADERO.

Tabla 43. Matriz de ST_Intersects (1). La función ST_Intersects devuelve un 1 (uno) si los interiores de ambas geometrías forman intersección.

	Interior Geometría b	**Perím. Geometría b**	**Exterior Geometría b**
Perím. Geometría a	*	*	*
Interior Geometría a	V	*	*
Exterior Geometría a	*	*	*

Tabla 44. Matriz de ST_Intersects (2). La función ST_Intersects devuelve un 1 (uno) si el perímetro de la primera geometría forma intersección con el perímetro de la segunda geometría.

	Interior Geometría b	**Perím. Geometría b**	**Exterior Geometría b**
Perím. Geometría a	*	*	*
Interior Geometría a	*	V	*
Exterior Geometría a	*	*	*

Tabla 45. Matriz de ST_Intersects (3). La función ST_Intersects devuelve un 1 (uno) si el perímetro de la primera geometría forma intersección con el interior de la segunda.

	Interior Geometría b	**Perím. Geometría b**	**Exterior Geometría b**
Perím. Geometría a	V	*	*
Interior Geometría a	*	*	*
Exterior Geometría a	*	*	*

Tabla 46. Matriz de ST_Intersects (4). La función ST_Intersects devuelve un 1 (uno) si los perímetros de cualquiera de las dos geometrías forma intersección.

	Interior Geometría b	**Perím. Geometría b**	**Exterior Geometría b**
Perím. Geometría a	*	V	*
Interior Geometría a	*	*	*
Exterior Geometría a	*	*	*

ST_Crosses

ST_Crosses toma dos geometrías como parámetros de entrada y devuelve un 1 (uno) si:

- La intersección da lugar a una geometría cuya dimensión es menor que la dimensión máxima de las geometrías originales.
- El conjunto de intersección es interior respecto a ambas geometrías originales.

ST_Crosses devuelve un valor nulo si la primera geometría es una superficie o multisuperficie o si la segunda geometría es un punto o multipunto. Para todas las demás combinaciones, ST_Crosses devuelve un valor de 1 (que indica que las dos geometrías se cruzan) o un valor de 0 (que indica que no se cruzan).

La siguiente figura muestra multipuntos que se cruzan con un cadena lineal, una cadena lineal que se cruza con una multilínea, varios puntos que se cruzan con un polígono y una cadena lineal que se cruza con un polígono. En tres de los cuatro casos, la geometría *b* se cruza con la geometría *a*. En el cuarto caso la geometría a es un multipunto que no cruza la línea pero sí que entra en contacto con el área interior del polígono de la geometría b.

Las geometrías en negro representan la geometría *a*; las geometrías en gris representan la geometría *b*.

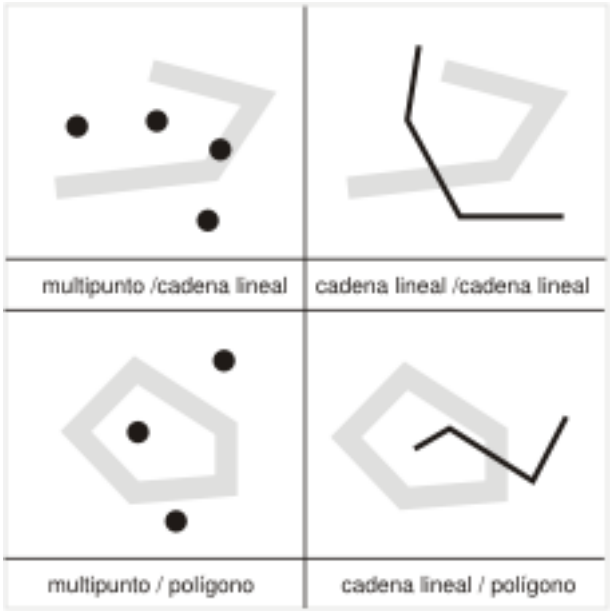


Figura 46. ST_Crosses

La matriz patrón de la Tabla 47 es aplicable si la primera geometría es un punto o multipunto, o si la primera geometría es una curva o multicurva y la segunda geometría es una superficie. La matriz indica que los interiores deben formar intersección y que el interior de la primera geometría (*a*) debe formar intersección con el exterior de la segunda geometría (*b*).

Tabla 47. Matriz de ST_Crosses (1)

	Interior Geometría b	**Perím. Geometría b**	**Exterior Geometría b**
Perím. Geometría a	*	*	*
Interior Geometría a	V	*	V
Exterior Geometría a	*	*	*

La matriz patrón de la Tabla 48 en la página 324 es aplicable si ambas geometrías son curvas o multicurvas. El 0 indica que la intersección de los interiores debe ser un punto (dimensión 0). Si la dimensión de esta intersección es 1 (intersección con un cadena lineal), la función ST_Crosses devuelve un 0 (que indica que las geometrías forman intersección); en cambio, la función ST_Overlaps devuelve un 1 (que indica que las geometrías se solapan).

Tabla 48. Matriz de ST_Crosses (2)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	*
Interior Geometría a	0	*	*
Exterior Geometría a	*	*	*

ST_Overlaps

ST_Overlaps compara dos geometrías de la misma dimensión. Esta función devuelve un 1 (uno) si el conjunto de intersección da como resultado una geometría que es diferente de las dos geometrías originales, pero que tiene la misma dimensión.

Las geometrías en negro representan la geometría *a*; las geometrías en gris representan la geometría *b*. En todos los casos, ambas geometrías tienen la misma dimensión y una se solapa parcialmente con la otra. El área de solapamiento es una nueva geometría, que tiene la misma dimensión que las geometrías *a* y *b*.

La siguiente figura ilustra el solapamiento de las geometrías. Los tres ejemplos muestran solapamientos con puntos, cadenas lineales y polígonos. Con puntos, los puntos en sí se solapan. Con cadenas lineales, una parte de la línea se solapa. Con polígonos, una parte del área se solapa.

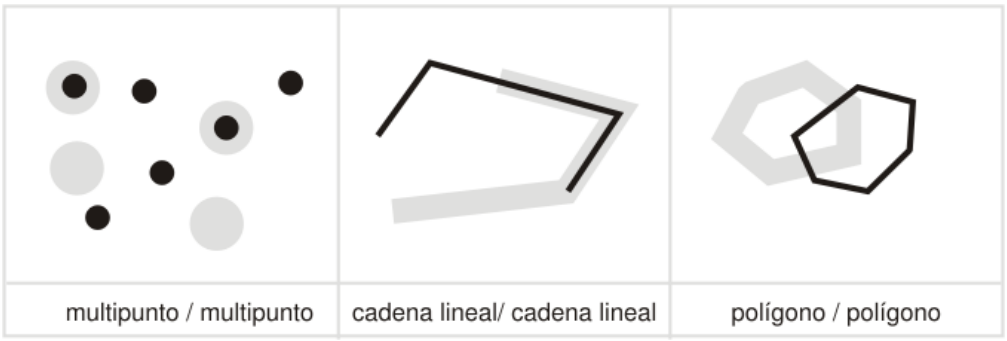


Figura 47. ST_Overlaps

La matriz patrón de la Tabla 49 es aplicable si ambas geometrías son puntos, multipuntos, superficies o multisuperficies. ST_Overlaps devuelve un 1 si el interior de cada geometría forma intersección con el interior y exterior de la otra geometría.

Tabla 49. Matriz de ST_Overlaps (1)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	*
Interior Geometría a	V	*	V
Exterior Geometría a	V	*	*

La matriz patrón de la Tabla 50 en la página 325 es aplicable si ambas geometrías son curvas o multicurvas. En este caso, la intersección de las geometrías debe dar lugar a una geometría cuya dimensión sea 1 (otra curva). Si la dimensión de la intersección de los interiores es 0, ST_Overlaps devuelve un valor de 0 (que indica que las geometrías no se solapan); en cambio, la función ST_Crosses devolvería un valor de 1 (que indica que las geometrías se cruzan).

Tabla 50. Matriz de ST_Overlaps (2)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	*
Interior Geometría a	1	*	V
Exterior Geometría a	V	*	*

ST_Touches

ST_devuelve un 1 (uno) si todos los puntos comunes a ambas geometrías se pueden encontrar sólo en los perímetros. Los interiores de las geometrías no deben formar intersección entre sí. Como mínimo una de las geometrías debe ser una curva, superficie, multicurva o multisuperficie.

Las geometrías en negro representan la geometría *a*; las geometrías en gris representan la geometría *b*. En todos los casos, el perímetro de la geometría *b* forma intersección con la geometría *a*. El interior de la geometría *b* permanece separado de la geometría *a*.

La figura siguiente muestra ejemplos de contactos entre distintos tipos de geometría, como por ejemplo punto y cadena lineal, cadena lineal y multilínea, punto y polígono, y cadena lineal y polígono.

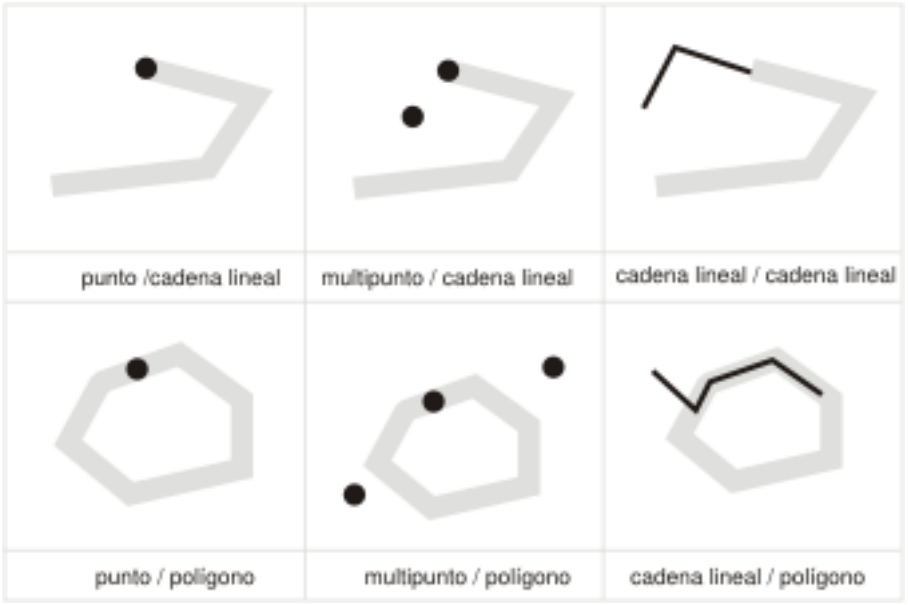


Figura 48. ST_Touches

Las matrices patrón muestran que la función ST_Touches devuelve un 1 (uno) cuando los interiores de la geometría no forman intersección, y el perímetro de cualquiera de las dos geometrías forma intersección con el interior o perímetro de la otra geometría.

Tabla 51. Matriz de ST_Touches (1)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	*	*
Interior Geometría a	F	V	*
Exterior Geometría a	*	*	*

Tabla 52. Matriz de ST_Touches (2)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	V	*	*
Interior Geometría a	F	*	*
Exterior Geometría a	*	*	*

Tabla 53. Matriz de ST_Touches (3)

	Interior Geometría b	Perím. Geometría b	Exterior Geometría b
Perím. Geometría a	*	V	*
Interior Geometría a	F	*	*
Exterior Geometría a	*	*	*

Funciones que comparan envolturas de geometrías

ST_EnvIntersects y ST_MBRIntersects son similares en cuanto que determinan si el rectángulo más pequeño que engloba a una geometría forma intersección con el rectángulo más pequeño que engloba a otra geometría. Hasta ahora, el término habitual utilizado para designar este rectángulo es el de *envoltura*. Los multipolígonos, polígonos, multilíneas y cadenas lineales curvadas se ajustan a los lados de sus envolturas; las cadenas lineales horizontales o verticales y los puntos son ligeramente menores que sus envolturas. ST_EnvIntersects determina si las envolturas de las geometrías forman intersección.

Para designar el área rectangular más pequeña en la que puede estar contenida una geometría se utiliza el término ″rectángulo delimitador mínimo″ (minimum bounding rectangle, MBR). Las envolturas que rodean a los multipolígonos, polígonos, multilíneas y cadenas lineales curvadas son en realidad un MBR. En cambio, las envolturas que rodean a cadenas lineales horizontales o verticales y a los puntos no son un MBR, pues no constituyen un área mínima donde puedan caber esas geometrías. Esas geometrías no ocupan ningún espacio definible y por tanto no pueden tener un MBR. Pero, por convenio, se considera que estas geometrías constituyen sus propios MBR. Por tanto, con respecto a multipolígonos, polígonos, multilíneas y cadenas lineales curvadas, ST_MBRIntersects verifica la condición de intersección para los mismos rectángulos delimitadores para los que lo hace ST_EnvIntersects. Excepto para las cadenas lineales horizontales y verticales y los puntos, ST_MBRIntersects verifica las intersecciones de las propias geometrías.

ST_EnvIntersects

ST_EnvIntersects devuelve un 1 (uno) si las envolturas de dos geometrías forman intersección. Es una función auxiliar que ejecuta de forma eficaz la función ST_Intersects (`ST_Envelope(g1),ST_Envelope(g2)`).

ST_MBRIntersects

ST_MBRIntersects devuelve un 1 (uno) si los rectángulos delimitadores mínimos (los MBR) de dos geometrías forman intersección.

Funciones que comprueban si dos elementos son idénticos

ST_EqualCoordsys

ST_EqualCoordsys devuelve un valor de 1 (uno) si dos definiciones de sistemas de coordenadas son idénticos. Cuando compara las definiciones, ST_EqualCoordsys

no tiene en cuenta las diferencias referentes al uso de mayúsculas/minúsculas, espacios, paréntesis y la representación de números de coma flotante.

ST_Equals

ST_Equals devuelve un 1 (uno) si dos geometrías son iguales. El orden de los puntos utilizados para definir las geometrías no afecta a la comprobación de igualdad.

En los seis ejemplos (punto, multipunto, cadena lineal, multilínea, polígono y multipolígono) geometría a y geometría b son las mismas.

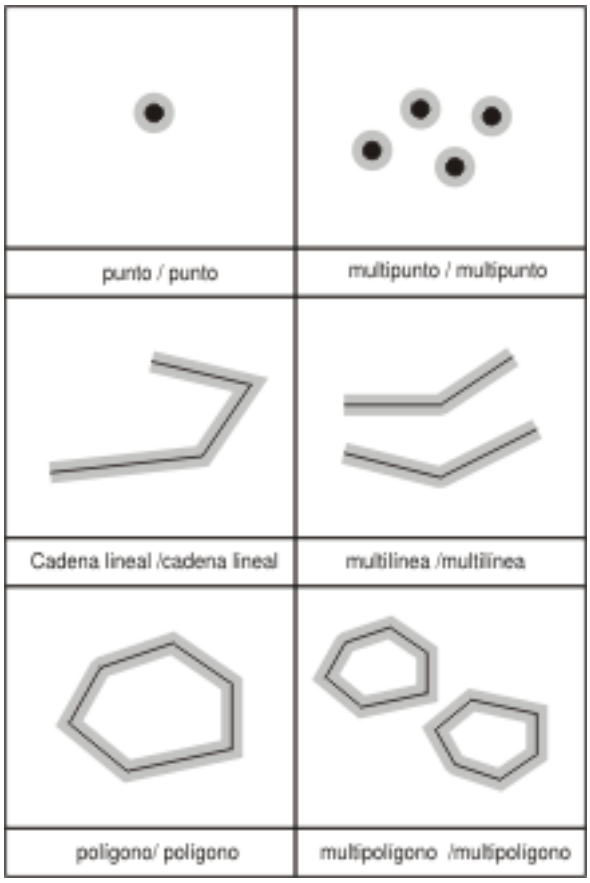


Figura 49. ST_Equals. Las geometrías en negro representan la geometría *a*; las geometrías en gris representan la geometría *b*. En todos los casos, la geometría *a* es igual a la geometría *b*.

Tabla 54. Matriz de igualdad. La matriz patrón DE-9IM para comprobar la condición de igualdad verifica que los interiores forman intersección y que ningún interior o perímetro de cualquiera de las geometrías forma intersección con el exterior de la otra.

	Interior Geometría b	**Perím. Geometría b**	**Exterior Geometría b**
Perím. Geometría a	*	*	F
Interior Geometría a	V	*	F
Exterior Geometría a	F	F	*

ST_EqualSRS

ST_EqualSRS devuelve un 1 (uno) si dos sistemas de referencia espacial son iguales, a condición de que el identificador numérico de uno o ambos sistemas no sea nulo.

Función que comprueba si no hay intersección entre dos geometrías

ST_Disjoint devuelve un 1 (uno) si la intersección de las dos geometrías es un conjunto vacío. El resultado devuelto por esta función es exactamente el opuesto al de ST_Intersects.

La ilustración muestra distintas geometrías y cómo los perímetros no forman intersección en ningún punto.

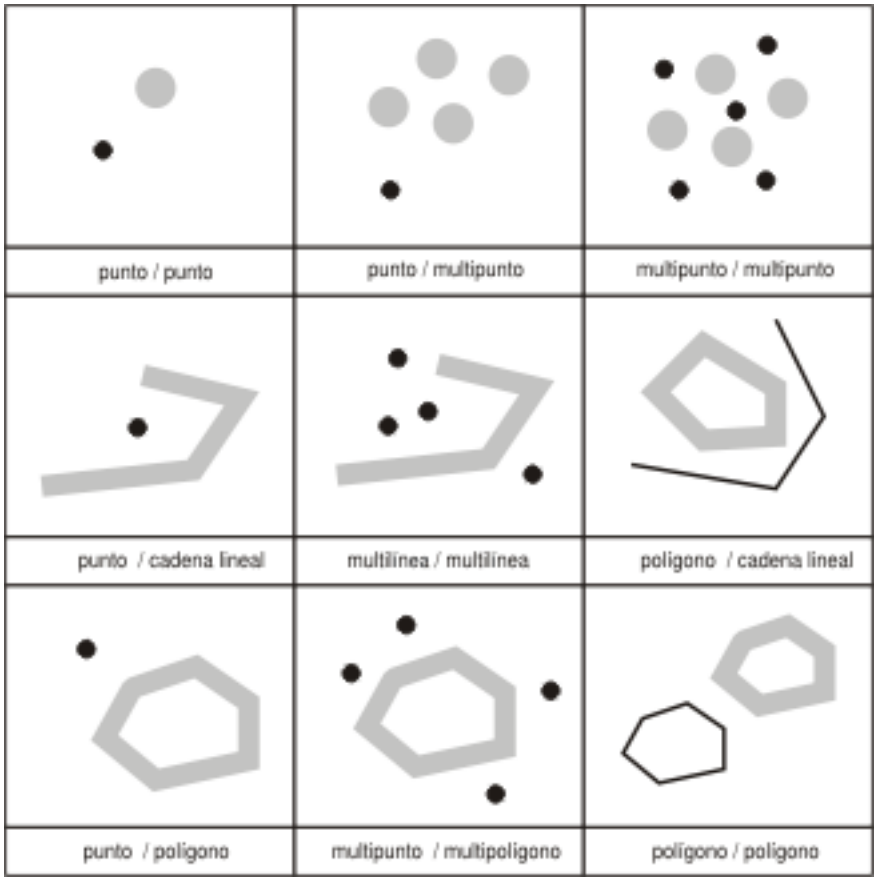


Figura 50. ST_Disjoint. Las geometrías en negro representan la geometría *a*; las geometrías en gris representan la geometría *b*. En todos los casos, la geometría *a* y la geometría *b* están inconexas la una respecto a la otra.

Tabla 55. Matriz de ST_Disjoint. Esta matriz indica que ni los interiores ni los perímetros de ninguna de las dos geometrías forma intersección.

	Interior Geometría b	**Perím. Geometría b**	**Exterior Geometría b**
Perím. Geometría a	F	F	*
Interior Geometría a	F	F	*
Exterior Geometría a	*	*	*

Función que compara geometrías con la serie de la matriz patrón DE-9IM

La función ST_Relate compara dos geometrías y devuelve un 1 (uno) si las geometrías cumplen las condiciones especificadas por la serie de la matriz patrón DE-9IM; en otro caso, la función devuelve un 0 (cero).

Funciones que proporcionan información sobre las propiedades de las geometrías

Este sección describe las funciones espaciales que devuelven información acerca de las propiedades de las geometrías. Esta información comprende:

- Tipos de datos de las geometrías
- Coordenadas y medidas dentro de una geometría
- Anillos, perímetros, envolturas y rectángulos delimitadores mínimos (los MBR)
- Dimensiones
- Indicación de si la geometría es cerrada, vacía o simple
- Geometrías base que componen una colección de geometrías
- Sistemas de referencia espacial

Algunas propiedades son geometrías por sí mismas; por ejemplo, los anillos exterior e interior de una superficie o los puntos inicial y final de una curva. Estas geometrías son producidas por algunas de las funciones incluidas en esta categoría. Las funciones que producen otras clases de geometrías; por ejemplo, las geometrías representativas de las zonas que circundan a un lugar determinado, pertenecen a otra categoría. Para obtener información sobre esta otra categoría, denominada "Funciones espaciales que generan nuevas geometrías", consulte el enlace o referencia cruzada apropiados al final de esta sección.

Conceptos relacionados:

- "Función que proporciona información sobre el tipo de datos" en la página 329
- "Funciones que proporcionan información sobre coordenadas y medidas" en la página 329
- "Funciones que proporcionan información sobre geometrías contenidas en una geometría" en la página 331
- "Funciones que proporcionan información sobre perímetros, envolturas y anillos" en la página 333
- "Funciones que proporcionan información sobre las dimensiones de una geometría" en la página 334
- "Funciones que indican si una geometría es cerrada, vacía o simple" en la página 335
- "Funciones que identifican el sistema de referencia espacial de una geometría" en la página 335

Función que proporciona información sobre el tipo de datos

ST_GeometryType toma una geometría como parámetro de entrada y devuelve el nombre de tipo calificado al completo del tipo dinámico de dicha geometría.

Funciones que proporcionan información sobre coordenadas y medidas

Las funciones siguientes devuelven información sobre las coordenadas y medidas de una geometría. Por ejemplo, ST_X devuelve la coordenada X de un punto especificado, ST_MaxX devuelve la coordenada X mayor de una geometría y ST_MinX devuelve la coordenada X menor de una geometría.

Estas funciones son:

- ST_CoordDim
- ST_IsMeasured
- ST_IsValid
- ST_Is3D
- ST_M
- ST_MaxM
- ST_MaxX
- ST_MaxY
- ST_MaxZ
- ST_MinM
- ST_MinX
- ST_MinY
- ST_MinZ
- ST_X
- ST_Y
- ST_Z

ST_CoordDim

ST_CoordDim devuelve un valor que indica qué tipos de coordenadas tiene una geometría y si la geometría también contiene alguna medida. Este valor se denomina *dimensión de coordenada*. Una dimensión de coordenada no es lo mismo que la propiedad denominada *dimensión*. Esta última indica si una geometría tiene un ancho o longitud, no si contiene coordenadas de un tipo específico o medidas.

ST_IsMeasured

ST_IsMeasured toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría tiene coordenadas M (medidas). En caso contrario, se devuelve un 0 (cero).

ST_IsValid

ST_IsValid toma una geometría como parámetro de entrada y devuelve un 1 si la geometría es válida. En caso contrario, se devuelve un 0 (cero). Una geometría es válida sólo si todos los atributos contenidos en el tipo estructurado son coherentes con la representación interna de los datos de la geometría, y si la representación interna no está dañada.

ST_Is3D

ST_Is3d toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría tiene coordenadas Z. En caso contrario, se devuelve un 0 (cero).

ST_M

Si un punto determinado contiene una medida, ST_M puede tomar el punto como parámetro de entrada y obtener la medida.

ST_MaxM

ST_MaxM toma una geometría como parámetro de entrada y devuelve su medida máxima.

ST_MaxX

ST_MaxX toma una geometría como parámetro de entrada y devuelve su coordenada X máxima.

ST_MaxY

ST_MaxY toma una geometría como parámetro de entrada y devuelve su coordenada Y máxima.

ST_MaxZ

ST_MaxZ toma una geometría como parámetro de entrada y devuelve su coordenada Z máxima.

ST_MinM

ST_MinM toma una geometría como parámetro de entrada y devuelve su medida mínima.

ST_MinX

ST_MinX toma una geometría como parámetro de entrada y devuelve la coordenada X mínima.

ST_MinY

ST_MinY toma una geometría como parámetro de entrada y devuelve su coordenada Y mínima.

ST_MinZ

ST_MinY toma una geometría como parámetro de entrada y devuelve su coordenada Z mínima.

ST_X

ST_X toma un punto como parámetro de entrada y devuelve la coordenada X del punto.

ST_Y

ST_Y toma un punto como parámetro de entrada y devuelve la coordenada Y del punto.

ST_Z

Si un punto determinado contiene una coordenada Z, ST_Z puede tomar el punto como parámetro de entrada y obtener la coordenada Z.

Funciones que proporcionan información sobre geometrías contenidas en una geometría

Las funciones siguientes devuelven información sobre geometrías contenidas en una geometría. Algunas funciones identifican puntos determinados contenidos en una geometría; otras obtienen el número de geometrías contenidas en una colección.

Estas funciones son:

- ST_Centroid

- ST_EndPoint
- ST_GeometryN
- ST_LineStringN
- ST_MidPoint
- ST_NumGeometries
- ST_NumLineStrings
- ST_NumPoints
- ST_NumPolygons
- ST_PointN
- ST_PolygonN
- ST_StartPoint

ST_Centroid

ST_Centroid toma una geometría como parámetro de entrada y devuelve el centro geográfico (el centro del rectángulo delimitador mínimo de la geometría) en forma de punto.

ST_EndPoint

ST_Endpoint toma una curva como parámetro de entrada y devuelve el punto que es el último punto de la curva.

ST_GeometryN

ST_GeometryN toma una colección de geometrías y un índice como parámetros de entrada y devuelve la geometría de la colección que está identificada por el índice.

ST_LineStringN

ST_LineStringN toma un multilínea y un índice como parámetros de entrada y devuelve la cadena lineal identificada por el índice.

ST_MidPoint

ST_MidPoint toma una curva como parámetro de entrada y devuelve un punto en la curva que es equidistante de los dos puntos finales de la curva, medido a lo largo de la curva.

ST_NumGeometries

ST_NumGeometries toma una colección de geometrías como parámetro de entrada y devuelve el número de geometrías de la colección.

ST_NumLineStrings

ST_NumLineStrings toma una multilínea como parámetro de entrada y devuelve el número de cadenas lineales de la multilínea.

ST_NumPoints

ST_NumPoints toma una geometría como parámetro de entrada y devuelve el número de puntos que se utilizaron para definir esa geometría. Por ejemplo, si la geometría es un polígono y se utilizaron cinco puntos para definir ese polígono, el número devuelto es 5.

ST_NumPolygons

ST_NumPolygons toma un multipolígono como parámetro de entrada y devuelve el número de polígonos que contiene.

ST_PointN

ST_PointN toma una cadena lineal o un multipunto, y un índice como parámetros de entrada y devuelve el punto identificado por el índice y perteneciente a la cadena lineal o multipunto.

ST_PolygonN

ST_PolygonN toma un multipolígono y un índice como parámetros de entrada y devuelve el polígono identificado por el índice.

ST_StartPoint

ST_StartPoint toma una curva como parámetro de entrada y devuelve el primer punto de la curva.

Funciones que proporcionan información sobre perímetros, envolturas y anillos

Las funciones siguientes devuelven información sobre demarcaciones que separan una parte interior de una geometría con respecto a una parte exterior, o que separan a la propia geometría con respecto al espacio que externo que la rodea. Por ejemplo, ST_Boundary devuelve el perímetro de una geometría representado en forma de curva.

Estas funciones son:

- ST_Boundary
- ST_Envelope
- ST_EnvIntersects
- ST_ExteriorRing
- ST_InteriorRingN
- ST_MBR
- ST_MBRIntersects
- ST_NumInteriorRing
- ST_Perimeter

ST_Boundary

ST_Boundary toma una geometría como parámetro de entrada y devuelve su perímetro en forma de nueva geometría.

ST_Envelope

ST_Envelope toma una geometría como parámetro de entrada y devuelve la envoltura de la geometría. La envoltura es un rectángulo que está representado por un polígono.

ST_EnvIntersects

ST_EnvIntersects toma dos geometrías como parámetros de entrada y devuelve un 1 si las envolturas de las dos geometrías forman intersección. En caso contrario, se devuelve un 0 (cero).

ST_ExteriorRing

ST_ExteriorRing toma un polígono como parámetro de entrada y devuelve su anillo exterior en forma de curva.

ST_InteriorRingN

ST_InteriorRingN toma un polígono y un índice como parámetros de entrada y devuelve el anillo interior identificado por dicho índice en forma de cadena lineal. Los anillos interiores se organizan según las normas definidas por las rutinas internas de verificación de la geometría.

ST_MBR

ST_MBR toma una geometría como parámetro de entrada y devuelve el rectángulo delimitador mínimo de la misma.

ST_MBRIntersects

ST_MBRIntersects devuelve un 1 (uno) si los rectángulos delimitadores mínimos (los MBR) de dos geometrías forman intersección.

ST_NumInteriorRing

ST_NumInteriorRing toma un polígono como parámetro de entrada y devuelve el número de sus anillos interiores.

ST_Perimeter

ST_Perimeter toma como parámetros de entrada una superficie o multisuperficie y, opcionalmente, una unidad de medida y devuelve el perímetro de la superficie o multisuperficie (es decir, la longitud de sus límites) expresado en la unidad de medida proporcionada.

Funciones que proporcionan información sobre las dimensiones de una geometría

Las funciones siguientes devuelven información sobre la dimensión de una geometría. Por ejemplo, ST_Area proporciona el área cubierta por una geometría determinada.

Estas funciones son:

- ST_Area
- ST_Dimension
- ST_Length

ST_Area

ST_Area toma como parámetros de entrada una geometría y, opcionalmente, una unidad de medida y devuelve el área cubierta por la geometría, expresada en la unidad de medida proporcionada.

ST_Dimension

ST_Dimension toma una geometría como parámetro de entrada y devuelve la dimensión del mismo.

ST_Length

ST_Length toma como parámetros de entrada una curva o multicurva y, opcionalmente, una unidad de medida y devuelve la longitud de la curva o multicurva proporcionada, expresada en la unidad de medida proporcionada.

Funciones que indican si una geometría es cerrada, vacía o simple

Las funciones siguientes indican si:

- Si una curva o multicurva determinada es cerrada (es decir, si el punto inicial y final de la curva o multicurva es el mismo punto)
- Si una geometría determinada está vacía (es decir, carece de puntos)
- Si una curva, multicurva o multipunto es simple (es decir, si tales geometrías tienen configuraciones típicas)

ST_IsClosed

ST_IsClosed toma una curva o multicurva como parámetro de entrada y devuelve un 1 si dicha curva o multicurva es cerrada. En caso contrario, se devuelve un 0 (cero).

ST_IsEmpty

ST_IsEmpty toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría está vacía. En caso contrario, se devuelve un 0 (cero).

ST_IsSimple

ST_IsSimple toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría es simple. En caso contrario, se devuelve un 0 (cero).

Funciones que identifican el sistema de referencia espacial de una geometría

Las funciones siguientes devuelven valores que identifican el sistema de referencia espacial correspondiente a una geometría. Además, la función ST_SrsID puede cambiar el sistema de referencia espacial de una geometría sin cambiar ni transformar la geometría.

ST_SrsId (también llamada ST_SRID)

ST_SrsId (o ST_SRID) toma como parámetros de entrada una geometría y, opcionalmente, un identificador de sistemas de referencia espacial. Los datos que devuelve esta función dependen de los parámetros de entrada que se hayan especificado:

- Si se especifica el identificador de sistemas de referencia espacial, la función devuelve la geometría con su identificador cambiado al identificador especificado. No se realiza ninguna transformación de la geometría.
- Si no se proporciona como parámetro de entrada ningún identificador de sistemas de referencia espacial, se devuelve el identificador actual de la geometría especificada.

ST_SrsName

ST_SrsName toma una geometría como parámetro de entrada y devuelve el nombre del sistema de referencia espacial en el que está representada dicha geometría.

Funciones que generan nuevas geometrías a partir de geometrías existentes

Este sección describe la categoría de funciones que obtienen nuevas geometrías a partir de otras existentes. Esta categoría no incluye las funciones que obtienen geometrías representativas de propiedades de otras geometrías. Sino que estas funciones realizan estas acciones:

- Convierten geometrías en otras geometrías
- Crean geometrías que representan configuraciones de espacio
- Obtienen geometrías individuales a partir de varias geometrías
- Crean geometrías basándose en medidas
- Crean modificaciones de geometrías

Conceptos relacionados:

- "Funciones que convierten una geometría en otra" en la página 336
- "Funciones que crean nuevas geometrías con configuraciones espaciales diferentes" en la página 337
- "Funciones que obtienen una geometría a partir de varias" en la página 341
- "Funciones que obtienen nuevas geometrías basándose en medidas" en la página 341
- "Funciones que crean formas modificadas de geometrías existentes" en la página 342

Funciones que convierten una geometría en otra

Las funciones siguientes pueden convertir geometrías pertenecientes a un supertipo en las geometrías correspondientes de un subtipo. Por ejemplo, la función ST_ToLineString puede convertir una cadena lineal de tipo ST_Geometry en una cadena lineal de tipo ST_LineString. Algunas de estas funciones pueden también combinar geometrías base y colecciones de geometrías para formar una colección de geometrías individual. Por ejemplo, ST_ToMultiLine puede convertir una cadena lineal y una multilínea en una multilínea individual.

ST_Polygon

ST_Polygon puede crear un polígono a partir de una cadena lineal cerrada. La cadena lineal definirá el anillo exterior del polígono.

ST_ToGeomColl

ST_ToGeomColl toma una geometría como parámetro de entrada y la convierte a un conjunto de geometrías.

ST_ToLineString

ST_ToLineString toma una geometría como parámetro de entrada y la convierte en una cadena lineal.

ST_ToMultiLine

ST_ToMultiLine toma una geometría como parámetro de entrada y la convierte en una multilínea.

ST_ToMultiPoint

ST_ToMultiPoint toma una geometría como parámetro de entrada y la convierte en multipunto.

ST_ToMultiPolygon

ST_ToMultiPolygon toma una geometría como parámetro de entrada y la convierte en un multipolígono.

ST_ToPoint

ST_ToPoint toma una geometría como parámetro de entrada y la convierte en un punto.

ST_ToPolygon

ST_ToPolygon toma una geometría como parámetro de entrada y la convierte en un polígono.

Funciones que crean nuevas geometrías con configuraciones espaciales diferentes

Utilizando geometrías existentes como punto de partida, las funciones siguientes crean nuevas geometrías que representan áreas circulares u otras configuraciones de espacio. Por ejemplo, dado un punto que representa el centro de un aeropuerto proyectado, ST_Buffer puede crear una superficie que represente, de forma circular, la extensión propuesta del aeropuerto.

Estas funciones son:

- ST_Buffer
- ST_ConvexHull
- ST_Difference
- ST_Intersection
- ST_SymDifference

ST_Buffer

La función ST_Buffer puede generar una nueva geometría que sobresale de una geometría existente de acuerdo con un radio especificado. La nueva geometría será una superficie cuando se cree una zona de separación en torno a la geometría existente o cuando los elementos de una colección están tan cerca unos de otros, que existe solapamiento de las zonas de separación que rodean a los elementos individuales de la colección. Sin embargo, cuando las zonas de separación están separadas, se originan superficies de separación individuales, en cuyo caso ST_Buffer devuelve una multisuperficie.

La figura siguiente ilustra la zona de separación situada alrededor de elementos únicos y solapados.

Figura 51. ST_Buffer

La función ST_Buffer acepta tanto una distancia positiva como negativa, pero sólo las geometrías cuya dimensión sea 2 (superficies y multisuperficies) aplican una zona de separación negativa. Se utiliza el valor absoluto de la distancia de separación cuando la dimensión de la geometría original es menor que 2 (todas las geometrías que no sean superficies ni multisuperficies).

En general, para los anillos exteriores, las distancias de separación positivas generan anillos de superficie que están separados del centro de la geometría original; las distancias de separación negativas generan anillos de superficie o multisuperficie dirigidos hacia el centro. Para los anillos interiores de una superficie o multisuperficie, una distancia de separación positiva genera un anillo de separación dirigido hacia el centro, y una distancia de separación negativa genera un anillo de separación separado del centro.

El proceso de creación de zonas de separación produce la fusión de las superficies que se solapan. Las distancias negativas mayores que la mitad del ancho interior máximo de un polígono dan lugar a una geometría vacía.

ST_ConvexHull

La función ST_ConvexHull devuelve la cáscara convexa de cualquier geometría que tenga como mínimo tres vértices formando una superficie convexa. Los *vértices* son pares de coordenadas X e Y situados dentro de las geometrías. Una *cáscara convexa* es el polígono convexo más pequeño que puede ser formado por todos los vértices de un conjunto determinado de vértices.

La siguiente ilustración muestra cuatro ejemplos de cáscara convexa. En el primer ejemplo se ha dibujado una forma irregular parecida a la letra c. La c está delimitada por la cáscara convexa. En el cuarto ejemplo hay cuatro puntos con líneas que siguen un patrón en forma de zigzag. La línea convexa discurre entre los puntos cuatro y dos en un lado y el tres y el uno en el otro lado.

Figura 52. ST_ConvexHull

ST_Difference

ST_Difference toma como entrada dos geometrías de la misma dimensión. La función ST_Difference devuelve la parte de la primera geometría que no forma intersección con la segunda geometría. Esta operación es el equivalente espacial de la operación lógica AND NOT. La porción de geometría devuelta por ST_Difference constituye por sí misma una geometría — una colección que tiene la misma dimensión que las geometrías utilizadas como entrada por la función. Si estas dos geometrías son iguales — es decir, ocupan el mismo espacio — la geometría resultante está vacía.

A la izquierda de cada flecha hay dos geometrías que se proporcionan a ST_Difference como entrada. A la derecha de cada flecha aparece la salida de ST_Difference. Si una parte de la primera geometría forma intersección con la segunda, el resultado de la función es esa parte de la primera geometría que no forma intersección. Si las geometrías proporcionadas como entrada son iguales, la salida resultante será una geometría vacía (indicada por el término *nil*)

Esta figura ilustra la entrada y la salida para ST_Difference. Por ejemplo, si la entrada son puntos, y el punto a y el punto b son iguales, la salida será un valor nulo. Si el punto a y el punto b son diferentes, la salida sería un nuevo punto situado entre los dos. Si la entrada fuera un polígono para b y un polígono más pequeño aunque idéntico para la geometría a situada dentro del primero, el resultado sería un valor nulo. Si los polígonos se solaparan, la salida se correspondería con los bordes exteriores de los polígonos combinados.

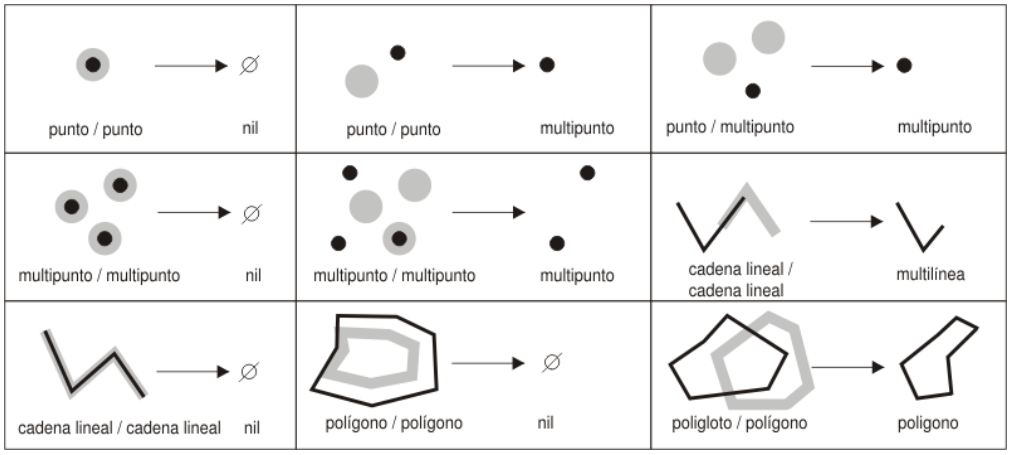


Figura 53. ST_Difference

ST_Intersection

La función ST_Intersection devuelve un conjunto de puntos, representados en forma de geometría, que definen la intersección de dos geometrías proporcionadas. Si las geometrías proporcionadas a ST_Intersection como entrada no forman

intersección, o bien si forman intersección y la dimensión de la misma es menor que las dimensiones de las geometrías, ST_Intersection devuelve una geometría vacía.

A la izquierda de cada flecha hay dos geometrías que forman intersección y se proporcionan a ST_Intersection como entrada. A la derecha de cada flecha aparece la salida de ST_Intersection, una geometría que representa la intersección creada por la geometrías de la izquierda.

Esta figura ilustra diez ejemplos de salida de ST_Intersection, que devuelve información sobre dónde forman intersección las geometrías. Por ejemplo, si b fuera una cadena lineal y la geometría a fuera un punto de la línea, la salida sería el multipunto en el que convergen la geometría a y la geometría b. Si la geometría a y la geometría b fueran polígonos solapados, la salida sería un nuevo multipolígono sólo con la porción solapada.

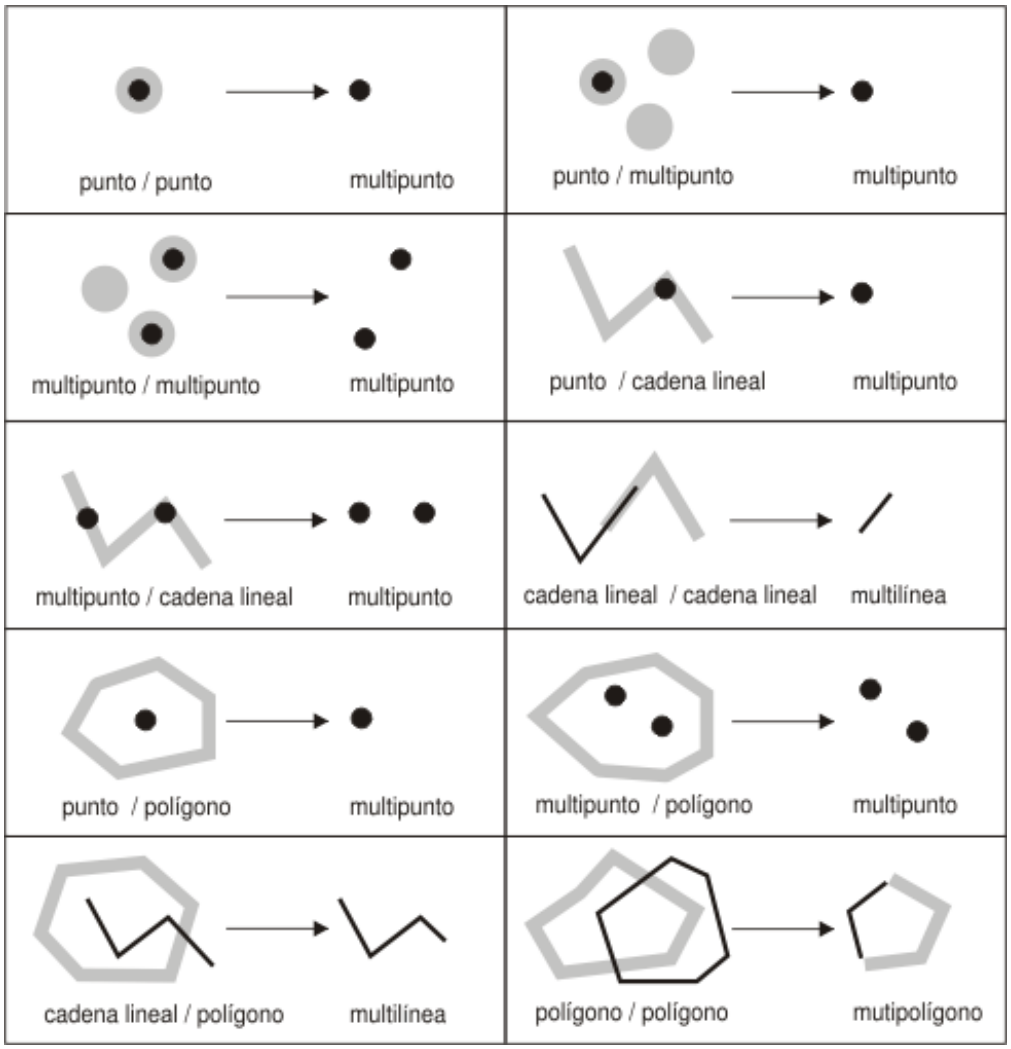


Figura 54. ST_Intersection

ST_SymDifference

La función ST_SymDifference devuelve la diferencia simétrica (el equivalente espacial de la operación lógica XOR) de dos geometrías que forman intersección y tienen la misma dimensión. Si estas geometrías son iguales, ST_SymDifference

devuelve una geometría vacía. Si las geometrías no son iguales, entonces una parte de una o ambas geometrías quedará fuera del área de intersección.

Funciones que obtienen una geometría a partir de varias

Las funciones siguientes obtienen geometrías individuales a partir de varias geometrías. Por ejemplo, ST_Union combina dos geometrías y forma una geometría individual.

MBR Aggregate (Agregado MBR)

El uso combinado de las funciones ST_BuildMBRAggr y ST_GetAggrResult agrega una columna de geometrías de una columna seleccionada a una geometría individual creando un rectángulo que representa el rectángulo delimitador mínimo (minimum bounding rectangle, MBR) que encierra todas las geometrías contenidas en la columna. Las coordenadas Z y M se descartan cuando se calcula el agregado.

ST_Union

La función ST_Union obtiene el conjunto de unión de dos geometrías. Esta operación es el equivalente espacial de la operación lógica OR. Las dos geometrías deben tener la misma dimensión. ST_Union siempre devuelve el resultado en forma de colección.

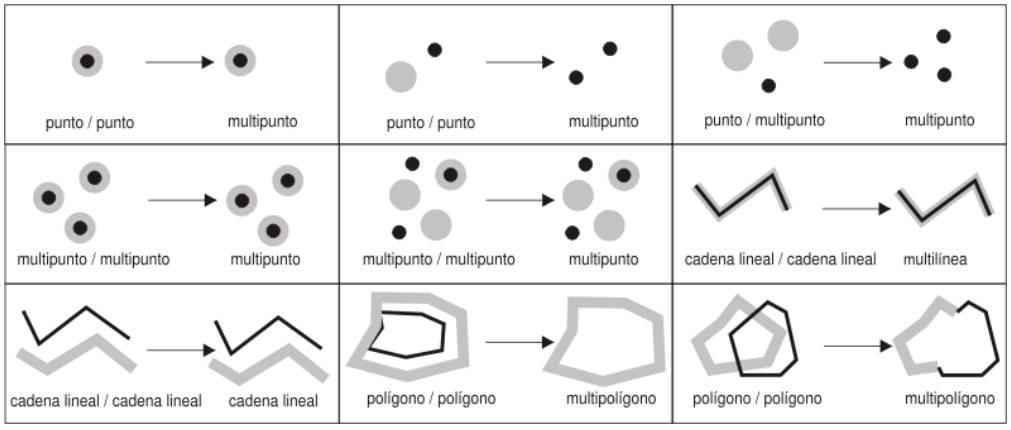


Figura 55. ST_Union

Union Aggregate

Un agregado de unión es la combinación de las funciones ST_BuildUnionAggr y ST_GetAggrResult. Esta combinación agrega una columna de geometrías de una tabla a una geometría individual mediante la creación de la unión.

Funciones que obtienen nuevas geometrías basándose en medidas

Las funciones tratadas en esta sección crean geometrías cuyos puntos tienen una medida o secuencia de dos medidas específica. Por ejemplo, suponga que los puntos de una multicurva contienen medidas cuyo valor está comprendido entre 4 y 8. Si desea conocer qué puntos tienen una medida igual a 7, puede utilizar la función ST_FindMeasure para obtener esos puntos en forma de un multipunto individual.

Estas funciones son:

- ST_FindMeasure (también llamada ST_LocateAlong)
- ST_MeasureBetween (también llamada ST_LocateBetween)

ST_FindMeasure (también llamada ST_LocateAlong)

ST_FindMeasure (o ST_LocateAlong) toma una geometría y una medida como parámetros de entrada. Devuelve un multipunto o una multicurva de la geometría en cuestión que coincide con la medida especificada. Para los puntos y multipuntos, el resultado es todos los puntos que tienen la medida especificada. En el caso de curvas, multicurvas, superficies y multisuperficies, se realiza una interpolación para calcular el resultado. Para las superficies y multisuperficies, el cálculo se realiza sobre el perímetro de la geometría.

ST_MeasureBetween (también llamada ST_LocateBetween)

ST_MeasureBetween (o ST_LocateBetween) toma como parámetros de entrada una geometría y dos coordenadas M (medidas) y devuelve la parte de la geometría proporcionada que representa el conjunto de puntos desconectados situados entre las dos coordenadas M.

En el caso de curvas, multicurvas, superficies y multisuperficies, se realiza una interpolación para calcular el resultado. En la Figura 56, los puntos 3, 4, 5, 6, 7, 8 y 9 representan una curva. Si las dos coordenadas M son el 4 y el 7, ST_MeasureBetween devuelve la parte de la curva situada entre los puntos 4 y 7.

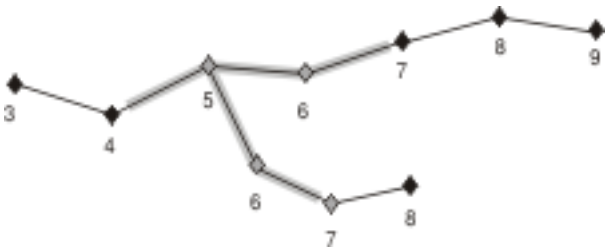


Figura 56. LocateBetween

Funciones que crean formas modificadas de geometrías existentes

Las funciones siguientes crean formas modificadas de geometrías existentes. Por ejemplo, la función ST_AppendPoint crea versiones ampliadas de curvas existentes. Cada versión contiene los puntos de una curva existente más un punto adicional.

Estas funciones son:

- ST_AppendPoint
- ST_ChangePoint
- ST_Generalize
- ST_M
- ST_PerpPoints
- ST_RemovePoint
- ST_X
- ST_Y
- ST_Z

ST_AppendPoint

ST_AppendPoint toma una curva y un punto como parámetros de entrada y extiende la curva por el punto determinado.

ST_ChangePoint

ST_ChangePoint toma una curva y dos puntos como parámetros de entrada. Sustituye todas las apariciones del primer punto de la curva de entrada por el segundo punto y devuelve la curva resultante.

ST_Generalize

ST_Generalize toma una geometría y un umbral como parámetros de entrada y representa dicha geometría con un número reducido de puntos, al mismo tiempo que conserva las características generales de la geometría. Se utiliza el algoritmo de simplificación de líneas de Douglas-Peucker, mediante el cual la secuencia de puntos que define la geometría se subdivide repetidamente hasta que un tramo de los puntos se puede sustituir por un segmento lineal recto. En este segmento lineal, ninguno de los puntos que lo definen se aparta del segmento lineal recto más allá del umbral especificado. Las coordenadas Z y M no se tienen en cuenta para la simplificación.

ST_M

Si un punto determinado no tiene una medida asociada a él, ST_M puede proporcionar una medida para guardarla con el punto. Si el punto tiene una medida asociada, ST_M puede sustituir esa medida por otra.

ST_PerpPoints

ST_PerpPoints toma como parámetros de entrada una curva o multicurva y un punto y devuelve la proyección perpendicular de ese punto en la curva o multicurva. Se obtiene el punto con la distancia más pequeña entre el punto proporcionado y el punto perpendicular. Si dos o más puntos proyectados perpendicularmente son equidistantes respecto al punto proporcionado, se devuelven todos ellos.

ST_RemovePoint

ST_RemovePoint toma una curva y un punto como parámetros de entrada y devuelve dicha curva después de eliminar de ella todos los puntos que son iguales al punto especificado. Si la curva proporcionada tiene coordenadas Z o M, el punto también debe tener coordenadas Z o M.

ST_X

ST_X puede sustituir la coordenada X de un punto por otra coordenada X.

ST_Y

ST_Y puede sustituir la coordenada Y de un punto por otra coordenada Y.

ST_Z

Si un punto proporcionado carece de coordenada Z, ST_Z puede añadir una coordenada Z al punto. Si el punto tiene una coordenada Z, ST_Z puede sustituir esta coordenada por otra coordenada Z.

Función que proporciona información sobre las distancias

ST_Distance toma como parámetros de entrada dos geometrías y, opcionalmente, una unidad de medida, y devuelve la distancia más corta entre cualquier punto de la primera geometría y cualquier punto de la segunda geometría, expresada en las unidades proporcionadas.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si cualquiera de las dos geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.

Por ejemplo, ST_Distance podría obtener la distancia más corta que un avión debe recorrer entre dos localidades. La Figura 57 muestra esta información devuelta.

La figura muestra un mapa de Estados Unidos con una línea recta entre dos puntos denominados Los Angeles y Chicago.

Figura 57. Distancia mínima entre dos ciudades. ST_Distance puede utilizar como entrada las coordenadas de Los Ángeles y Chicago y obtener el valor correspondiente a la distancia mínima entre esas localidades.

Función que proporciona información sobre el índice

ST_GetIndexParms toma como parámetro de entrada el identificador de un índice espacial o columna espacial y devuelve los parámetros utilizados para definir el índice en la columna espacial. Opcionalmente se puede especificar un número de parámetro, en cuyo caso sólo se obtiene el parámetro identificado por el número.

Conversiones entre sistemas de coordenadas

ST_Transform toma como parámetros de entrada una geometría y un identificador de sistemas de referencia espacial y transforma la geometría para que esté representada en el sistema de referencia espacial especificado. Se realizan proyecciones y conversiones entre diferentes sistemas de coordenadas y se ajustan las coordenadas de las geometrías según convenga.

Capítulo 23. Funciones espaciales: sintaxis y parámetros

Esta sección presenta las funciones espaciales descritas en las secciones siguientes. Trata acerca de algunos factores que son comunes para todas o para la mayoría de funciones espaciales. A continuación se documentan las funciones en orden alfabético.

Funciones espaciales: consideraciones y tipos de datos asociados

Esta sección describe lo que necesita conocer para codificar funciones espaciales. Esta información comprende:

- Factores a tener en cuenta: la necesidad de especificar el esquema al que pertenecen las funciones espaciales y el hecho de que puede invocar a algunas funciones como métodos.
- Cómo abordar una situación en la que una función espacial no puede procesar el tipo de geometría devuelto por otra función espacial.
- Una tabla que muestra una lista de las funciones espaciales de acuerdo con el tipo de dato espacial utilizado como entrada

Factores a tener en cuenta

Cuando utilice funciones espaciales, tenga en cuenta estos factores:

- Para poder invocar a una función espacial, el nombre de la función debe estar calificado por el nombre del esquema al que pertenecen las funciones espaciales: DB2GSE. Una forma de hacer esto es especificar explícitamente el esquema en la sentencia de SQL donde se utiliza la función; por ejemplo:

  ```
  SELECT db2gse.ST_Relate (g1, g2, 'T*F**FFF2') EQUALS FROM relate_test
  ```

  Como método alternativo, para evitar especificar el esquema cada vez que se debe llamar a una función, puede añadir DB2GSE al registro especial CURRENT FUNCTION PATH. Para obtener los valores actuales de este registro especial, escriba este mandato de SQL:

  ```
  VALUES CURRENT FUNCTION PATH
  ```

  Para actualizar el registro especial CURRENT FUNCTION PATH con DB2GSE, emita este mandato de SQL:

  ```
  set CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
  ```

- Puede invocar a algunas funciones espaciales como métodos. Por ejemplo, en el código siguiente, se invoca a ST_Area primero como función y luego como método. En ambos casos, ST_Area está codificado para actuar sobre un polígono cuyo ID es 10 y que está contenido en la columna SALES_ZONE de la tabla STORES. Cuando se invoca a ST_Area, ésta devuelve el área del elemento geográfico real —Sales Zone no. 10— representado por el polígono.

  ST_Area invocada como función:

  ```
  SELECT ST_Area(sales_zone)
  FROM   stores
  WHERE  id = 10
  ```

  ST_Area invocada como método:

  ```
  SELECT sales_zone..ST_Area()
  FROM   stores
  WHERE  id = 10
  ```

Tratamiento de valores de ST_Geometry como valores de un subtipo

Si una función espacial devuelve una geometría cuyo tipo estático es un supertipo y la geometría se pasa a una función que sólo acepta geometrías de un tipo que esté subordinado a este supertipo, se emite una excepción de compilación.

Por ejemplo, el tipo estático del parámetro de salida de la función ST_Union es ST_Geometry, el supertipo de todos los tipos de datos espaciales. El parámetro de entrada estático de la función ST_PointOnSurface puede ser ST_Polygon o ST_MultiPolygon, dos subtipos de ST_Geometry. Si DB2® intenta pasar a ST_PointOnSurface geometrías devueltas por ST_Union, DB2 emite esta excepción de compilación:

```
SQL00440N No se ha encontrado en la vía de función ninguna función llamada
"ST_POINTONSURFACE" que tenga argumentos compatibles. SQLSTATE=42884
```

Este mensaje indica que DB2 no pudo encontrar una función llamada ST_PointOnSurface y que tenga un parámetro de entrada de tipo ST_Geometry.

Para permitir que las geometrías de un supertipo pasen a funciones que sólo aceptan subtipos del supertipo, utilice el operador TREAT. Tal como se indicó anteriormente, ST_Union devuelve geometrías cuyo tipo estático es ST_Geometry. También puede devolver geometrías cuyo subtipo dinámico es ST_Geometry. Por ejemplo, suponga que la función devuelve una geometría cuyo tipo dinámico es ST_MultiPolygon. En ese caso, el operador TREAT necesita que esta geometría se utilice con el tipo estático ST_MultiPolygon. Esto coincide con uno de los tipos de datos del parámetro de entrada de ST_PointOnSurface. Si ST_Union no devuelve un valor ST_MultiPolygon, DB2 emite una excepción de ejecución.

Si una función devuelve una geometría de un supertipo, normalmente el operador TREAT puede indicar a DB2 que trate esta geometría como un subtipo de ese supertipo. Pero tenga en cuenta que esta operación solo es efectiva si el subtipo coincide con o está subordinado a un subtipo estático definido como parámetro de entrada de la función a la que se pasa la geometría. Si no se cumple esta condición, DB2 emite una excepción de ejecución.

Considere otro ejemplo: suponga que desea determinar los puntos perpendiculares para un punto determinado situado en el límite de un polígono que carece de huecos. Puede utilizar la función ST_Boundary para obtener el límite a partir del polígono. El parámetro de salida estático de ST_Boundary es ST_Geometry, pero ST_PerpPoints acepta geometrías ST_Curves. Debido a que todos los polígonos tienen como límite una cadena lineal (que es también una curva), y debido a que el tipo de datos Cadena Lineal (ST_LineString) está subordinado a ST_Curve, la operación siguiente permitirá que un polígono ST_Geometry devuelto por ST_Boundary se pase a ST_PerpPoints:

```
SELECT ST_AsText(ST_PerpPoints(TREAT(ST_Boundary(polygon) as ST_Curve)),
       ST_Point(30.5, 65.3, 1)))
FROM   polygon_table
```

En lugar de invocar a ST_Boundary y ST_PerpPoints como funciones, puede invocarlos como métodos. Para ello, especifique este código:

```
SELECT TREAT(ST_Boundary(polygon) as ST_Curve)..
       ST_PerpPoints(St_Point(30.5, 65.3, ))..ST_AsText()
FROM   polygon_table
```

Funciones espaciales listadas según el tipo de datos de entrada

La Tabla 56 lista las funciones espaciales de acuerdo con el tipo de datos de entrada que pueden aceptar.

Importante: Tal como se indicó en otro lugar, los tipos de datos espaciales forman una jerarquía, cuya raíz es ST_Geometry. Cuando la documentación de DB2 Spatial Extender indica que un valor de un supertipo de esta jerarquía se puede utilizar como entrada para una función, como alternativa también se puede utilizar como entrada un valor de cualquier subtipo de ese supertipo.

Por ejemplo, las primeras de entradas de la Tabla 56 indican que ST_Area y varias otras funciones pueden utilizar como entrada valores del tipo de datos ST_Geometry. Por tanto, estas funciones también pueden utilizar como entrada valores de cualquier subtipo de ST_Geometry: ST_Point, ST_Curve, ST_LineString, etc.

Tabla 56. Funciones espaciales de acuerdo con el tipo de datos de entrada

Tipo de datos del parámetro de entrada	Función
ST_Geometry	EnvelopesIntersect ST_Area ST_AsBinary ST_AsGML ST_AsShape ST_AsText ST_Boundary ST_Buffer ST_BuildMBRAggr ST_BuildUnionAggr ST_Centroid ST_Contains ST_ConvexHull ST_CoordDim ST_Crosses ST_Difference ST_Dimension ST_Disjoint ST_Distance ST_Envelope ST_EnvIntersects ST_Equals ST_FindMeasure o ST_LocateAlong ST_Generalize ST_GeometryType

Tabla 56. Funciones espaciales de acuerdo con el tipo de datos de entrada (continuación)

Tipo de datos del parámetro de entrada	Función
ST_Geometry (continuación)	ST_Intersection ST_Intersects ST_Is3D ST_IsEmpty ST_IsMeasured ST_IsSimple ST_IsValid ST_MaxM ST_MaxX ST_MaxY ST_MaxZ ST_MBR ST_MBRIntersects ST_MeasureBetween o ST_LocateBetween ST_MinM ST_MinX ST_MinY ST_MinZ ST_NumPoints ST_Overlaps ST_Relate ST_SRID o ST_SrsId ST_SrsName ST_SymDifference ST_ToGeomColl ST_ToLineString ST_ToMultiLine ST_ToMultiPoint ST_ToMultiPolygon ST_ToPoint ST_ToPolygon ST_Touches ST_Transform ST_Union ST_Within
ST_Point	ST_M ST_X ST_Y ST_Z
ST_Curve	ST_AppendPoint ST_ChangePoint ST_EndPoint ST_IsClosed ST_IsRing ST_Length ST_MidPoint ST_PerpPoints ST_RemovePoint ST_StartPoint
ST_LineString	ST_PointN ST_Polygon
ST_Surface	ST_Perimeter ST_PointOnSurface
ST_GeomCollection	ST_GeometryN ST_NumGeometries

Tabla 56. Funciones espaciales de acuerdo con el tipo de datos de entrada (continuación)

Tipo de datos del parámetro de entrada	Función
ST_MultiPoint	ST_PointN
ST_MultiCurve	ST_IsClosed ST_Length ST_PerpPoints
ST_MultiLineString	ST_LineStringN ST_NumLineStrings ST_Polygon
ST_MultiSurface	ST_Perimeter ST_PointOnSurface
ST_MultiPolygon	ST_NumPolygons ST_PolygonN

Las funciones ST_BuildMBRAggr y ST_BuildUnionAggr se describen en ″MBR Aggregate″ (Agregado MBR) y ″Union Aggregate″ (Agregado de unión), respectivamente.

Información relacionada:

- “MBR Aggregate (Agregado MBR)” en la página 353
- “ST_Boundary” en la página 364
- “ST_Area” en la página 356
- “ST_PerpPoints” en la página 476
- “ST_Point” en la página 478
- “ST_PointOnSurface” en la página 484
- “ST_Relate” en la página 491
- “ST_Union” en la página 510
- “Union aggregate (Agregado de unión)” en la página 519

EnvelopesIntersect

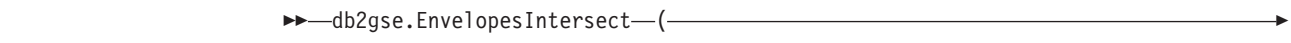
EnvelopesIntersect acepta dos tipos de parámetros de entrada:

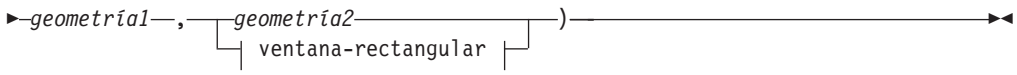
- Dos geometrías

  EnvelopesIntersect devuelve un 1 si la envoltura de la primera geometría forma una intersección con la envoltura de la segunda geometría. En caso contrario, se devuelve un 0 (cero).
- Una geometría, cuatro valores de coordenadas de tipo DOUBLE que definen los ángulos inferior izquierdo y superior derecho de una ventana rectangular y el identificador de sistemas de referencia espacial.

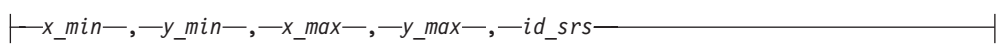
  EnvelopesIntersect devuelve un 1 si la envoltura de la primera geometría forma una intersección con la envoltura definida por los cuatro valores de tipo DOUBLE. En caso contrario, se devuelve un 0 (cero).

Sintaxis:





ventana-rectangular:



Parámetros:

geometría1
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría cuya envoltura se debe verificar si forma intersección con la envoltura de *geometría2* o la ventana rectangular definida por los cuatro valores de tipo DOUBLE.

geometría2
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría cuya envoltura se debe verificar si forma intersección con la envoltura de *geometría1*.

x_min
Especifica el valor mínimo de la coordenada X para la envoltura. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

Para los datos geodésicos se aplican las siguientes normas:

- *x_min* debe ser un valor de longitud comprendido entre –180 y 180 grados.
- *x_min* es mayor que *x_max* cuando la envoltura se solapa con el meridiano 180º.

y_min
Especifica el valor mínimo de la coordenada Y para la envoltura. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

Para los datos geodésicos se aplican las siguientes normas:

- *y_min* debe ser un valor de latitud comprendido entre –90 y 90 grados.
- *y_min* debe ser menor que el valor *y_max*.

x_max
Especifica el valor máximo de la coordenada X para la envoltura. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

Para los datos geodésicos se aplican las siguientes normas:

- *x_max* debe ser un valor de longitud comprendido entre –180 y 180 grados.
- *x_max* es menor que el valor *x_min* cuando la envoltura se solapa con el meridiano 180º.

y_max
Especifica el valor máximo de la coordenada Y para la envoltura. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es DOUBLE.

Para los datos geodésicos se aplican las siguientes normas:

- *y_max* debe ser un valor de latitud comprendido entre –90 y 90 grados.

- *y_max* debe ser mayor que el valor *y_min*.

id_srs
Identifica de forma exclusiva el sistema de referencia espacial. El identificador de sistemas de referencia espacial debe coincidir con el identificador de sistemas de referencia espacial del parámetro de geometría. Debe especificar un valor que no sea nulo para este parámetro.

El tipo de datos de este parámetro es INTEGER.

Tipo de retorno:

INTEGER

Ejemplo:

En este ejemplo se crean dos polígonos que representan condados y se determina si alguno de ellos forma una intersección con un área geográfica especificada por los cuatro valores de tipo DOUBLE.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE condados (id INTEGER, nombre CHAR(20), geometría ST_Polygon)

INSERT INTO condados VALUES
       (1, 'Condado_1', ST_Polygon('polígono((0 0, 30 0, 40 30, 40 35,
        5 35, 5 10, 20 10, 20 5, 0 0))' ,0))

INSERT INTO condados VALUES
       (2, 'Condado_2', ST_Polygon('polígono((15 15, 15 20, 60 20, 60 15,
        15 15))' ,0))

INSERT INTO condados VALUES
       (3, 'Condado_3', ST_Polygon('polígono((115 15, 115 20, 160 20, 160 15,
        115 15))' ,0))


SELECT nombre
FROM condados as c
WHERE EnvelopesIntersect(c.geometry, 15, 15, 60, 20, 0) =1
```

Resultados:

```
Nombre--------------------
Condado_1
Condado_2
```

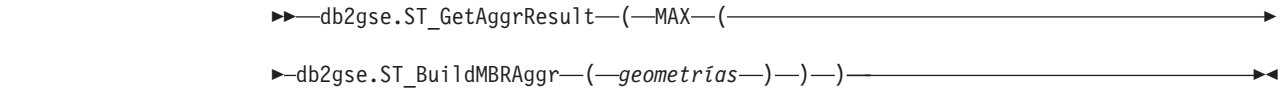
MBR Aggregate (Agregado MBR)

El uso combinado de las funciones ST_BuildMBRAggr y ST_GetAggrResult agrega una columna de geometrías de una columna seleccionada a una geometría individual creando un rectángulo que representa el rectángulo delimitador mínimo (minimum bounding rectangle, MBR) que encierra todas las geometrías contenidas en la columna. Las coordenadas Z y M se descartan cuando se calcula el agregado.

Si todas las geometrías que se deben combinar son nulas, se devuelve un valor nulo. Si todas las geometrías son nulas o están vacías, el resultado es una geometría vacía. Si el rectángulo delimitador mínimo de todas las geometrías a combinar da como resultado un punto, se devuelve este punto como valor de tipo ST_Point. Si el rectángulo delimitador mínimo de todas las geometrías a combinar da como resultado una cadena lineal horizontal o vertical, se devuelve esta cadena

lineal como valor de tipo ST_LineString. En otro caso, el rectángulo delimitador mínimo se devuelve como valor de tipo ST_Polygon.

Sintaxis:



Parámetro:

geometrías
Es una columna seleccionada que tiene un tipo de ST_Geometry o uno de sus subtipos y representa todas las geometrías para las que se debe calcular el rectángulo delimitador mínimo.

Tipo de retorno:

db2gse.ST_Geometry

Restricciones:

No puede crear el agregado de unión de una columna espacial en una selección completa en los casos siguientes:

- En un entorno MPP.
- Si se utiliza la cláusula GROUP BY en la selección completa.
- Si utiliza una función distinta de la función de agregación MAX de DB2.

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar la función ST_BuildMBRAggr para obtener el rectángulo delimitador máximo de todas las geometrías contenidas en una columna. Este ejemplo añade varios puntos a la columna GEOMETRY de la tabla SAMPLE_POINTS. Luego, el código de SQL determina el rectángulo delimitador máximo de todos los puntos puestos juntos.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_points (id integer, geometry ST_Point)

INSERT INTO sample_points (id, geometry)
VALUES
    (1, ST_Point(2,   3, 1)),
    (2, ST_Point(4,   5, 1)),
    (3, ST_Point(13, 15, 1)),
    (4, ST_Point(12,  5, 1)),
    (5, ST_Point(23,  2, 1)),
    (6, ST_Point(11,  4, 1))


SELECT cast(ST_GetAggrResult(MAX(ST_BuildMBRAggr
    (geometry)))..ST_AsText AS varchar(160))
    AS ";Aggregate_of_Points";
FROM sample_points
```

Resultados:

```
Aggregate_of_Points
----------------------------------------------------------
POLYGON (( 2.00000000 2.00000000, 23.00000000 2.00000000,
23.00000000 15.00000000, 2.00000000 15.00000000, 2.00000000
2.00000000))
```

ST_AppendPoint

ST_AppendPoint toma una curva y un punto como parámetros de entrada y extiende la curva por el punto determinado. Si la curva proporcionada tiene coordenadas Z o M, el punto también debe tener coordenadas Z o M. La curva resultante se representa según el sistema de referencia espacial de la curva proporcionada.

Si el punto a añadir no está representado según el mismo sistema de referencia espacial que la curva, se convertirá al otro sistema de referencia espacial.

Si la curva proporcionada es cerrada o simple, es posible que la curva resultante ya no sea cerrada ni simple. Si la curva o el punto proporcionado tiene un valor nulo, o si la curva está vacía, se devuelve un valor nulo. Si el punto que se debe añadir está vacío, se devuelve inalterada la curva de entrada y se emite un aviso (SQLSTATE 01HS3).

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

curva Valor de tipo ST_Curve o uno de sus subtipos que representa la curva a la que se añade *punto*.

punto Valor de tipo ST_Point que representa el punto que se añade a la *curva*.

Tipo de retorno:

db2gse.ST_Curve

Ejemplos:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este código crea dos cadenas lineales, cada una con tres puntos.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines(id integer, line ST_Linestring)

INSERT INTO sample_lines VALUES
    (1, ST_LineString('linestring (10 10, 10 0, 0 0 )', 0) )

INSERT INTO sample_lines VALUES
    (2, ST_LineString('linestring z (0 0 4, 5 5 5, 10 10 6)', 0) )
```

Ejemplo 1:

Este ejemplo añade el punto (5, 5) al final de una cadena lineal.

```
SELECT CAST(ST_AsText(ST_AppendPoint(line, ST_Point(5, 5)))
    AS VARCHAR(120)) New
FROM   sample_lines
WHERE  id=1
```

Resultados:

```
NEW
------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 10.00000000 0.00000000,
0.00000000 0.00000000, 5.00000000 5.00000000)
```

Ejemplo 2:

Este ejemplo añade el punto (15, 15, 7) al final de una cadena lineal con coordenadas Z.

```
SELECT CAST(ST_AsText(ST_AppendPoint(line, ST_Point(15.0, 15.0, 7.0)))
   AS VARCHAR(160)) New
FROM   sample_lines
WHERE  id=2
```

Resultados:

```
NEW
------------------------------------------------------------------
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 5.00000000
5.00000000 5.00000000, 10.00000000 10.00000000 6.00000000,
15.00000000 15.00000000 7.00000000)
```

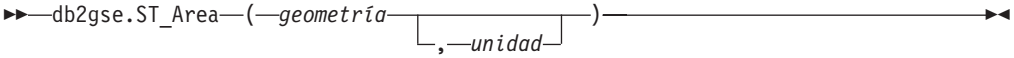
ST_Area

ST_Area toma como parámetros de entrada una geometría y, opcionalmente, una unidad de medida, y devuelve el área cubierta por la geometría, expresada en la unidad de medida proporcionada.

Si la geometría es un polígono o un multipolígono, se devuelve el área cubierta por la geometría. El área correspondiente a puntos, cadenas lineales, multipuntos y multilíneas es 0 (cero). Si la geometría es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que define el área.

unidad Valor de tipo VARCHAR(128) que identifica las unidades utilizadas para medir el área. Las unidades de medida soportadas están listadas en la vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE.

Si se omite el parámetro *unidad*, se siguen las normas siguientes para determinar la unidad utilizada para medir el área:

- Si *geometría* está en un sistema de coordenadas proyectadas o geocéntricas, se utiliza la unidad lineal correspondiente a este sistema de coordenadas.
- Si la *geometría* está en un sistema de coordenadas geográficas pero no en un sistema de referencia espacial (SRS) geodésico, se utiliza la unidad angular correspondiente a este sistema de coordenadas.
- Si la *geometría* está en un SRS geodésico, la unidad de medida por omisión será el metro cuadrado.

Restricciones para las conversiones de unidades: Se devuelve un error (SQLSTATE 38SU4) si se cumple alguna de las siguientes condiciones:

- La geometría está en un sistema de coordenadas no especificado y se especifica el parámetro *unidad*.
- La geometría está en un sistema de coordenadas proyectadas y se especifica una unidad angular.
- La geometría está en un sistema de coordenadas geográficas pero no en un SRS geodésico y se ha especificado una unidad lineal.
- La geometría está en un sistema de coordenadas geográficas, está en un SRS geodésico y se especifica una unidad angular.

Tipo de retorno:

DOUBLE

Ejemplos:

Ejemplo 1:

Necesitamos conocer el área cubierta por cada región de ventas de la empresa. Los polígonos correspondientes a las regiones de ventas están contenidos en la tabla SAMPLE_POLYGONS. El área se calcula aplicando la función ST_Area a la columna de geometrías.

```
db2se create_srs se_bank  -srsId 4000 -srsName new_york1983 -xOffset 0
    -yOffset 0 -xScale 1 -yScale 1
    -coordsysName NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet

CREATE TABLE sample_polygons (id INTEGER, geometry ST_POLYGON)

INSERT INTO sample_polygons (id, geometry)
VALUES
  (1, ST_Polygon('polygon((0 0, 0 10, 10 10, 10 0, 0 0))', 4000) ),
  (2, ST_Polygon('polygon((20 0, 30 20, 40 0, 20 0 ))', 4000) ),
  (3, ST_Polygon('polygon((20 30, 25 35, 30 30, 20 30))', 4000))
```

La sentencia SELECT siguiente recupera el ID y el área de la región de ventas:

```
SELECT id, ST_Area(geometry) AS area
FROM   sample_polygons
```

Resultados:

```
ID        AREA
--------- ------------------------
        1   +1.00000000000000E+002
        2   +2.00000000000000E+002
        3   +2.50000000000000E+001
```

Ejemplo 2:

La sentencia SELECT siguiente recupera el ID y el área de la región de ventas expresada en diversas unidades:

```
SELECT id,
       ST_Area(geometry) square_feet,
       ST_Area(geometry, 'METER') square_meters,
       ST_Area(geometry, 'STATUTE MILE') square_miles
FROM   sample_polygons
```

Resultados:

```
ID  SQUARE_FEET              SQUARE_METERS            SQUARE_MILES
--- ------------------------ ------------------------ ------------------------
  1 +1.00000000000000E+002   +9.29034116132748E+000   +3.58702077598427E-006
  2 +2.00000000000000E+002   +1.85806823226550E+001   +7.17404155196855E-006
  3 +2.50000000000000E+001   +2.32258529033187E+000   +8.96755193996069E-007
```

Ejemplo 3:

Este ejemplo determina el área de un polígono que está definido mediante coordenadas de Plano de Estado.

Con el siguiente mandato se crea el sistema de referencia espacial de Plano de estado con un ID de 3:

```
db2se create_srs SAMP_DB  -srsId 3 -srsName z3101a -xOffset 0
    -yOffset 0 -xScale 1 -yScale 1
    -coordsysName NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet
```

Las siguientes sentencias de SQL añaden el polígono del sistema de referencia espacial 3 a la tabla y determinan el área en pies cuadrados, metros cuadrados y millas cuadradas.

```
SET current function path db2gse;
 CREATE TABLE Sample_Poly3 (id integer, geometry ST_Polygon);
 INSERT into Sample_Poly3 VALUES
    (1, ST_Polygon('polygon((567176.0 1166411.0,
                             567176.0 1177640.0,
                             637948.0 1177640.0,
                             637948.0 1166411.0,
                             567176.0 1166411.0 ))', 3));
 SELECT id, ST_Area(geometry) "Square Feet",
            ST_Area(geometry, 'METER') "Square Meters",
            ST_Area(geometry, 'STATUTE MILE') "Square Miles"
 FROM Sample_Poly3;
```

Resultados:

```
ID  Pies cuadrados           Metros cuadrados         Millas cuadradas
--- ------------------------ ------------------------ ------------------------
  1 +7.94698788000000E+008   +7.38302286101346E+007   +2.85060106320552E+001
```

Ejemplo 4:

El analista espacial necesita una lista del área cubierta por cada región de exploración. Los polígonos de región de exploración se almacenan en la tabla SAMPLE_GEODETIC_TAB e incluyen las siguientes regiones:

- Una región alrededor del polo norte
- Una región alrededor del polo sur
- Una región que ocupa el meridiano 180º

El segundo campo del siguiente archivo de entrada, samp_wkt_rows.txt, contiene polígonos que representan estas regiones:

```
1|'polygon((5 82,15 82,25 82,35 82,45 82,55 82,65 82,75 82,85 82,95 82,
105 82,115 82,125 82,135 82,145 82,155 82,165 82,175 82,-175 82,-165 82,
-155 82,-145 82,-135 82,-125 82,-115 82,-105 82,-95 82,-85 82,-75 82,
-65 82,-55 82,-45 82,-35 82,-25 82,-15 82,-5 82,5 82))'|'North Pole region'
2|'polygon((175 -82,165 -82,155 -82,145 -82,135 -82,125 -82,115 -82,
105 -82,95 -82,85 -82,75 -82,65 -82,55 -82,45 -82,35 -82,25 -82,15 -82,
5 -82,-5 -82,-15 -82,-25 -82,-35 -82,-45 -82,-55 -82,-65 -82,-75 -82,
-85 -82,-95 -82,-105 -82,-115 -82,-125 -82,-135 -82,-145 -82,-155 -82,
-165 -82,-175 -82,175 -82))'|'South Pole region'
3|'polygon((-175 -42,-175 1,-175 42,175 42,175 -1,175 -42,-175 -42))
'|'180th meridian'
```

Las siguientes sentencias de SQL añaden los polígonos del sistema de referencia espacial geodésico 2000000000 a la tabla SAMPLE_GEODETIC_TAB.

```
SET current function path db2gse;
CREATE TABLE db2se_samp.gsege_temp_samp (
            gid      INTEGER,
            g1_wkt   varchar(500),
            comment  varchar(255)
   ) NOT LOGGED INITIALLY;
LOAD FROM samp_wkt_rows.txt OF DEL MODIFIED BY CHARDEL'' COLDEL|
   INSERT INTO db2se_samp.gsege_temp_samp;

CREATE TABLE sample_geodetic_tab
  (gid INTEGER NOT NULL PRIMARY KEY,
   geometry ST_Geometry),
   comment varchar(255));

INSERT INTO sample_geodetic_tab
     SELECT gid, ST_GeomFromText(g1_wkt, 2000000000), comment
   FROM db2se_samp.gsege_temp_samp;
```

La función ST_Area calcula el área del polígono en la columna de geometría. La unidad de medida por omisión de ST_Area es el metro cuadrado. La siguiente sentencia SELECT recupera el ID y el área de la región de exploración expresada en metros cuadrados, pies cuadrados y millas cuadradas.

```
SELECT id, ST_Area(geometry) AS SQUARE_METERS,
ST_Area(geometry,'FOOT') AS SQUARE_FEET,
ST_Area(geometry, 'STATUTE MILE') AS SQUARE_MILES
FROM sample_geodetic_tab
WHERE id BETWEEN 1 AND 9 ORDER BY id;
```

```
ID  SQUARE_METERS           SQUARE_FEET             SQUARE_MILES
--- ----------------------  ----------------------  ----------------------
  1 +2.52472719957839E+012  +2.71759374028922E+013  +9.74802621488040E+005
  2 +2.52475431563494E+012  +2.71762292776957E+013  +9.74813091056005E+005
  3 +9.43568029137069E+012  +1.01564817377028E+014  +3.64313652781464E+006
```

Información relacionada:

- “Funciones espaciales soportadas por DB2 Geodetic Extender” en la página 215
- “Vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE” en la página 306

ST_AsBinary

ST_AsBinary toma una geometría como parámetro de entrada y devuelve su presentación binaria convencional. Las coordenadas Z y M se descartan y no estarán representadas en la representación binaria convencional.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que se convertirá a la representación binaria convencional.

Tipo de retorno:

BLOB(2G)

Ejemplos:

El código siguiente muestra cómo utilizar la función ST_AsBinary para convertir los puntos contenidos en las columnas de geometrías de la tabla SAMPLE_POINTS en una representación binaria convencional (WKB) en la columna BLOB.

```
CREATE TABLE SAMPLE_POINTS (id integer, geometry ST_POINT, wkb BLOB(32K))

INSERT INTO SAMPLE_POINTS (id, geometry)
VALUES
    (1100, ST_Point(10, 20, 1))
```

Ejemplo 1:

Este ejemplo llena la columna binaria (WKB), cuyo ID es 1111, a partir de la columna GEOMETRY, cuyo ID es 1100.

```
INSERT INTO sample_points(id, wkb)
VALUES (1111,
  (SELECT  ST_AsBinary(geometry)
FROM   sample_points
   WHERE   id = 1100))

SELECT id, cast(ST_Point(wkb)..ST_AsText AS varchar(35)) AS point
FROM   sample_points
WHERE  id = 1111
```

Resultados:

```
ID          Point
----------- -----------------------------------
       1111 POINT ( 10.00000000 20.00000000)
```

Ejemplo 2:

Este ejemplo muestra la representación binaria WKB.

```
SELECT id, substr(ST_AsBinary(geometry), 1, 21) AS point_wkb
FROM   sample_points
WHERE  id = 1100
```

Resultados:

```
ID     POINT_WKB
------ ----------------------------------------
  1100 x'0101000000000000000000244000000000000003440'
```

Información relacionada:

- "Representación binaria convencional (WKB)" en la página 534

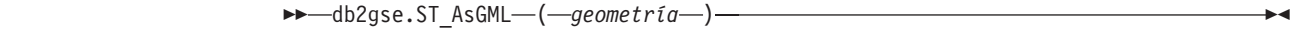
ST_AsGML

ST_AsGML toma una geometría como parámetro de entrada y devuelve su representación utilizando el lenguaje GML (Geography Markup Language).

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que se convertirá en la correspondiente representación GML.

Tipo de retorno:

CLOB(2G)

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El fragmento de código siguiente muestra cómo utilizar la función ST_AsGML para visualizar el fragmento GML. Este ejemplo llena la columna GML, a partir de la columna de geometrías cuyo ID es 2222.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE SAMPLE_POINTS (id integer, geometry ST_POINT, gml CLOB(32K))

INSERT INTO SAMPLE_POINTS (id, geometry)
VALUES
    (1100, ST_Point(10, 20, 1))

INSERT INTO sample_points(id, gml)
VALUES (2222,
  (SELECT  ST_AsGML(geometry)
FROM   sample_points
   WHERE   id = 1100))
```

La sentencia SELECT siguiente lista el ID y la representación GML de las geometrías. La geometría se convierte en un fragmento GML mediante la función ST_AsGML.

```
SELECT id, cast(ST_AsGML(geometry) AS varchar(110)) AS gml_fragment
FROM   sample_points
WHERE  id = 1100
```

Resultados:

```
La sentencia SELECT devuelve el resultado siguiente:

ID          GML_FRAGMENT

----------- ------------------------------------------------------------
       1100 <gml:Point srsName";EPSG:4269";><gml:coord>
            <gml:X>10</gml:X><gml:Y>20</gml:Y>
            </gml:coord></gml:Point>
```

Información relacionada:

- "Funciones espaciales que convierten valores de geometría en formatos de intercambio de datos" en la página 307
- "Representación GML (Geography Markup Language)" en la página 536

ST_AsShape

ST_AsShape toma una geometría como parámetro de entrada y devuelve su presentación de forma ESRI.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que se convertirá a la representación de forma ESRI correspondiente.

Tipo de retorno:

BLOB(2G)

Ejemplo:

El código siguiente muestra cómo utilizar la función ST_AsShape para convertir los puntos contenidos en la columna de geometrías de la tabla SAMPLE_POINTS en una representación binaria de formas en la columna BLOB de formas. Este ejemplo llena la columna de formas a partir de la columna de geometrías. La representación binaria de formas se utiliza para visualizar las geometrías en geonavegadores, los cuales necesitan que las geometrías se ajusten al formato del archivo de formas ESRI, o para crear las geometrías para el archivo *.SHP del archivo de formas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE SAMPLE_POINTS (id integer, geometry ST_POINT, shape BLOB(32K))

INSERT INTO SAMPLE_POINTS (id, geometry)
VALUES
    (1100, ST_Point(10, 20, 1))

INSERT INTO sample_points(id, shape)
VALUES (2222,
  (SELECT  ST_AsShape(geometry)
FROM   sample_points
   WHERE   id = 1100))

SELECT id, substr(ST_AsShape(geometry), 1, 20) AS shape
FROM   sample_points
WHERE  id = 1100
```

Resultado:

```
ID     SHAPE

------ --------------------------------------------
  1100 x'0100000000000000000024400000000000003440'
```

Información relacionada:

- "Representación de forma" en la página 536

ST_AsText

ST_AsText toma una geometría como parámetro de entrada y devuelve su representación de texto convencional.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
: Valor de tipo ST_Geometry o uno de sus subtipos que se convertirá a la representación de texto convencional correspondiente.

Tipo de retorno:

CLOB(2G)

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Después de capturar e insertar los datos en la tabla SAMPLE_GEOMETRIES, el analista verifica la corrección de los valores insertados examinando la representación de texto convencional de las geometrías.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id SMALLINT, spatial_type varchar(18),
    geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, spatial_type, geometry)
VALUES
    (1, 'st_point', ST_Point(50, 50, 0)),
    (2, 'st_linestring', ST_LineString('linestring
        (200 100, 210 130, 220 140)',  0)),
    (3, 'st_polygon', ST_Polygon('polygon((110 120, 110 140,
        130 140, 130 120, 110 120))', 0))
```

La sentencia SELECT siguiente lista el tipo espacial y la representación de texto convencional (WKT) de las geometrías. La geometría se convierte a texto mediante la función ST_AsText. Luego se convierte al tipo varchar(120) debido a que los datos de salida de la función ST_AsText son de tipo CLOB(2G).

```
SELECT id, spatial_type, cast(geometry..ST_AsText
   AS varchar(150)) AS wkt
FROM   sample_geometries
```

Resultados:

```
ID     SPATIAL_TYPE       WKT
------ ------------------ -------------------------------------------------
     1 st_point           POINT ( 50.00000000 50.00000000)

     2 st_linestring      LINESTRING ( 200.00000000 100.00000000,
                            210.00000000 130.00000000, 220.00000000
                            140.00000000)

     3 st_polygon         POLYGON (( 110.00000000 120.00000000,
                            130.00000000 120.00000000, 130.00000000
                            140.00000000, 110.00000000140.00000000,
                            110.00000000 120.00000000))
```

Información relacionada:

• "Representación de texto convencional (WKT)" en la página 529

ST_Boundary

ST_Boundary toma una geometría como parámetro de entrada y devuelve su perímetro en forma de nueva geometría. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si la geometría proporcionada es un punto, un multipunto, una curva cerrada o una multicurva cerrada, o si está vacía, el resultado es una geometría vacía de tipo ST_Point. Paras las curvas o multicurvas que no son cerradas, los puntos de inicio y final de las curvas se proporcionan en forma de valor de tipo ST_MultiPoint, a menos que dicho punto sea el punto inicial o final de un número par de curvas. Para las superficies y multisuperficies, se obtiene la curva que define el perímetro de la geometría proporcionada, en forma de valor de tipo ST_Curve o ST_MultiCurve. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

Si es posible, el tipo específico de la geometría devuelta será ST_Point, ST_LineString o ST_Polygon. Por ejemplo, el perímetro de un polígono sin huecos es una cadena lineal individual, representada como ST_LineString. El perímetro de un polígono con uno o varios huecos consiste en varias cadenas lineales, representadas como ST_MultiLineString.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos. Se devuelve el perímetro de esta geometría.

Tipo de retorno:

db2gse.ST_Geometry

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo crea varias geometrías y determina el perímetro de cada geometría.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (1, ST_Polygon('polygon((40 120, 90 120, 90 150, 40 150, 40 120))', 0))

INSERT INTO sample_geoms VALUES
       (2, ST_Polygon('polygon((40 120, 90 120, 90 150, 40 150, 40 120),
                               (70 130, 80 130, 80 140, 70 140, 70 130))' ,0))

INSERT INTO sample_geoms VALUES
       (3, ST_Geometry('linestring(60 60, 65 60, 65 70, 70 70)' ,0))

INSERT INTO sample_geoms VALUES
       (4, ST_Geometry('multilinestring((60 60, 65 60, 65 70, 70 70),
                                        (80 80, 85 80, 85 90, 90 90),
                                        (50 50, 55 50, 55 60, 60 60))' ,0))

INSERT INTO sample_geoms VALUES
       (5, ST_Geometry('point(30 30)' ,0))

SELECT id, CAST(ST_AsText(ST_Boundary(geometry)) as VARCHAR(320)) Boundary
FROM   sample_geoms

Resultados
ID      BOUNDARY
------- ------------------------------------------------------------------
      1 LINESTRING ( 40.00000000 120.00000000, 90.00000000 120.00000000,
          90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
          120.00000000)

      2 MULTILINESTRING (( 40.00000000 120.00000000, 90.00000000 120.00000000,
          90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
          120.00000000),( 70.00000000 130.00000000, 70.00000000 140.00000000,
          80.00000000 140.00000000, 80.00000000 130.00000000, 70.00000000
          130.00000000))

      3 MULTIPOINT ( 60.00000000 60.00000000, 70.00000000 70.00000000)

      4 MULTIPOINT ( 50.00000000 50.00000000, 70.00000000 70.00000000,
          80.00000000 80.00000000, 90.00000000 90.00000000)

      5 POINT EMPTY
```

ST_Buffer

ST_Buffer toma como parámetros de entrada una geometría, una distancia y, opcionalmente, una unidad de medida y devuelve una geometría que engloba a la geometría proporcionada y está separada de ella por la distancia especificada, expresada en la unidad indicada. Cada punto del perímetro de la geometría resultante está separado de la geometría proporcionada por la distancia especificada. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

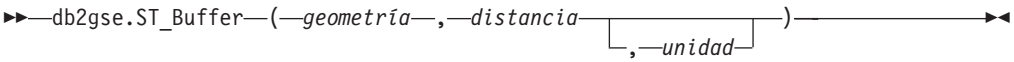
Para los datos geodésicos, si especifica una distancia negativa, ST_Buffer devuelve una región que va más allá de la distancia especificada de todos los puntos de la geometría de entrada. En otras palabras, una distancia negativa devuelve la región complementaria.

Los tramos curvos del perímetro de la geometría resultante se representan mediante cadenas lineales. Por ejemplo, la zona de separación alrededor de un punto, que daría lugar a una región circular, se representa mediante un polígono cuyo perímetro es una cadena lineal.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la cual se debe crear la zona de separación. Para los datos geodésicos, ST_Buffer sólo soporta los tipos de datos ST_Point y ST_MultiPoint.

distancia
Valor de tipo DOUBLE PRECISION que especifica la distancia que se debe utilizar para la zona de separación que debe haber alrededor de *geometría*. Para los datos geodésicos, la distancia no debe superior a la del radio ecuatorial de la tierra. Para el elipsoide WGS-84, esta longitud es de 6378137.0 metros.

unidad Valor de tipo VARCHAR(128) que identifica la unidad utilizada para medir la *distancia*. Las unidades de medida soportadas están listadas en la vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE.

Si se omite el parámetro *unidad*, se siguen las normas siguientes para determinar la unidad utilizada para medir la distancia:

- Si la *geometría* está en un sistema de coordenadas proyectadas o geocéntricas, se utiliza la unidad lineal correspondiente a este sistema de coordenadas por omisión.

- Si la *geometría* está en un sistema de coordenadas geográficas pero no en un sistema de referencia espacial (SRS) geodésico (SRS), se utiliza la unidad angular correspondiente a este sistema de coordenadas por omisión.
- Si la *geometría* está en un SRS geodésico, la unidad de medida por omisión será el metro.

Restricciones para las conversiones de unidades: Se devuelve un error (SQLSTATE 38SU4) si se cumple alguna de las siguientes condiciones:

- La geometría está en un sistema de coordenadas no especificado y se especifica el parámetro *unidad*.
- La geometría está en un sistema de coordenadas proyectadas y se especifica una unidad angular.
- La geometría está en un sistema de coordenadas geográficas pero no en un SRS geodésico y se ha especificado una unidad lineal.
- La geometría está en un sistema de coordenadas geográficas, está en un SRS geodésico y se especifica una unidad angular.

Tipo de retorno:

db2gse.ST_Geometry

Ejemplos:

En los ejemplos siguientes, los resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente crea un sistema de referencia espacial, y crea y llena la tabla SAMPLE_GEOMETRIES.

```
db2se create_srs se_bank  -srsId 4000 -srsName new_york1983
    -xOffset 0 -yOffset 0 -xScale 1 -yScale 1
    -coordsysName NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE
    sample_geometries (id INTEGER, spatial_type varchar(18),
    geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, spatial_type, geometry)
VALUES
    (1, 'st_point', ST_Point(50, 50, 4000)),
    (2, 'st_linestring',
         ST_LineString('linestring(200 100, 210 130,
         220 140)',  4000)),
    (3, 'st_polygon',
         ST_Polygon('polygon((110 120, 110 140, 130 140,
         130 120, 110 120))',4000)),
    (4, 'st_multipolygon',
         ST_MultiPolygon('multipolygon(((30 30, 30 40,
         35 40, 35 30, 30 30),(35 30, 35 40, 45 40,
         45 30, 35 30)))', 4000))
```

Ejemplo 1:

La sentencia SELECT siguiente utiliza la función ST_Buffer para aplicar una zona de separación cuyo valor es 10.

```
SELECT id, spatial_type,
       cast(geometry..ST_Buffer(10)..ST_AsText AS varchar(470)) AS buffer_10
FROM   sample_geometries
```

Resultados:

```
ID          SPATIAL_TYPE       BUFFER_10
----------- ------------------ ----------------------------------------
1           st_point           POLYGON (( 60.00000000 50.00000000,
   59.00000000 55.00000000, 54.00000000 59.00000000, 49.00000000
   60.00000000, 44.00000000 58.00000000, 41.00000000 53.00000000,
   40.00000000 48.00000000,42.00000000 43.00000000, 47.00000000
   41.00000000, 52.00000000 40.00000000, 57.00000000 42.00000000,
   60.00000000 50.00000000))

2             st_linestring    POLYGON (( 230.00000000
   140.00000000, 229.00000000 145.00000000, 224.00000000
   149.00000000, 219.00000000 150.00000000, 213.00000000 147.00000000,
   203.00000000 137.00000000, 201.00000000 133.00000000, 191.00000000
   103.00000000, 191.00000000 99.00000000, 192.00000000  95.00000000,
   196.00000000 91.00000000, 200.00000000 91.00000000,204.00000000
   91.00000000, 209.00000000 97.00000000, 218.00000000 124.00000000,
   227.00000000 133.00000000, 230.00000000 140.00000000))

3              st_polygon         POLYGON (( 140.00000000 120.00000000,
   140.00000000 140.00000000, 139.00000000  145.00000000, 130.00000000
   150.00000000, 110.00000000 150.00000000, 105.00000000 149.00000000,
   100.00000000 140.00000000,100.00000000 120.00000000, 101.00000000
   115.00000000, 110.00000000 110.00000000,130.00000000 110.00000000,
   135.00000000 111.00000000, 140.00000000 120.00000000))

4              st_multipolygon    POLYGON (( 55.00000000 30.00000000,
   55.00000000 40.00000000, 54.00000000 45.00000000, 45.00000000
   50.00000000, 30.00000000 50.00000000, 25.00000000 49.00000000,
   20.00000000 40.00000000, 20.00000000 30.00000000, 21.00000000
   25.00000000, 30.00000000 20.00000000, 45.00000000 20.00000000,
   50.00000000 21.00000000, 55.00000000 30.00000000))
```

Ejemplo 2:

La sentencia SELECT siguiente utiliza la función ST_Buffer para aplicar una zona de separación negativa cuyo valor es 5.

```
SELECT id, spatial_type,
       cast(ST_AsText(ST_Buffer(geometry, -5)) AS varchar(150))
       AS buffer_negative_5
FROM   sample_geometries
WHERE  id = 3
```

Resultados:

```
ID          SPATIAL_TYPE       BUFFER_NEGATIVE_5
----------- ------------------ ------------------------------------
3           st_polygon         POLYGON (( 115.00000000 125.00000000,
   125.00000000 125.00000000, 125.00000000 135.00000000, 115.00000000
   135.00000000, 115.00000000 125.00000000))
```

Ejemplo 3:

La sentencia SELECT siguiente muestra el resultado de aplicar un almacenamiento intermedio con el parámetro de unidad especificado.

```
SELECT id, spatial_type,
    cast(ST_AsText(ST_Buffer(geometry, 10, 'METER')) AS varchar(680))
    AS buffer_10_meter
FROM   sample_geometries
WHERE  id = 3
```

Resultados:

```
ID          SPATIAL_TYPE       BUFFER_10_METER
----------- ------------------ -------------------------------------
3           st_polygon         POLYGON (( 163.00000000 120.00000000,
   163.00000000 140.00000000, 162.00000000 149.00000000, 159.00000000
   157.00000000, 152.00000000 165.00000000, 143.00000000 170.00000000,
   130.00000000 173.00000000, 110.00000000 173.00000000, 101.00000000
   172.00000000, 92.00000000 167.00000000, 84.00000000 160.00000000,
   79.00000000 151.00000000, 77.00000000 140.00000000, 77.00000000
   120.00000000, 78.00000000 111.00000000, 83.00000000 102.00000000,
   90.00000000 94.00000000, 99.00000000 89.00000000, 110.00000000
   87.00000000, 130.00000000 87.00000000, 139.00000000 88.00000000,
   147.00000000 91.00000000, 155.00000000 98.00000000, 160.00000000
   107.00000000, 163.00000000 120.00000000))
```

Información relacionada:

- "Funciones espaciales soportadas por DB2 Geodetic Extender" en la página 215
- "Vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE" en la página 306

ST_Centroid

ST_Centroid toma una geometría como parámetro de entrada y devuelve el centro geográfico (el centro del rectángulo delimitador mínimo de la geometría) en forma de punto. El punto resultante se representa utilizando el sistema de referencia espacial de la geometría proporcionada.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la cual se debe determinar el centro geométrico.

Tipo de retorno:

db2gse.ST_Point

Ejemplo:

Este ejemplo crea dos geometrías y encuentra el centroide de las mismas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
```

```
    (1, ST_Polygon('polygon
    ((40 120, 90 120, 90 150, 40 150, 40 120),
    (50 130, 80 130, 80 140, 50 140, 50 130))',0))

INSERT INTO sample_geoms VALUES
    (2, ST_MultiPoint('multipoint(10 10, 50 10, 10 30)' ,0))


SELECT id, CAST(ST_AsText(ST_Centroid(geometry))
   as VARCHAR(40)) Centroid
FROM sample_geoms
```

Resultados:

```
ID          CENTROID
----------- ----------------------------------------
          1 POINT ( 65.00000000 135.00000000)
          2 POINT ( 30.00000000 20.00000000)
```

ST_ChangePoint

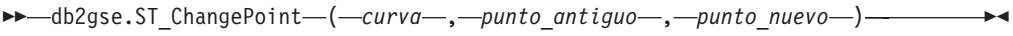
ST_ChangePoint toma una curva y dos puntos como parámetros de entrada. Sustituye todas las apariciones del primer punto de la curva de entrada por el segundo punto y devuelve la curva resultante. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si los dos puntos no están representados en el mismo sistema de referencia espacial que la curva, se convertirán al sistema de referencia espacial utilizado para la curva.

Si la curva de entrada está vacía, el resultado es un valor vacío. Si la curva de entrada es un valor nulo, o cualquiera de los puntos de entrada es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

curva Valor de tipo ST_Curve o un de sus subtipos que representa la curva en la que los puntos identificados como *punto_antiguo* se cambian a *punto_nuevo*.

punto_antiguo
Valor de tipo ST_Point que identifica los puntos de la curva que se cambian a *punto_nuevo*.

punto_nuevo
Valor de tipo ST_Point que representa las nuevas ubicaciones de los puntos de la curva identificados por *punto_antiguo*.

Tipo de retorno:

db2gse.ST_Curve

Restricciones:

El punto que se debe cambiar en la curva debe ser uno de los puntos utilizados para definir la curva.

Si la curva tiene coordenadas Z o M, dichos puntos también deben tener coordenadas Z o M.

Ejemplos:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente crea y llena la tabla SAMPLE_LINES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines(id INTEGER, line ST_Linestring)


INSERT INTO sample_lines VALUES
    (1, ST_LineString('linestring (10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0) )

INSERT INTO sample_lines VALUES
    (2, ST_LineString('linestring z (0 0 4, 5 5 5, 10 10 6, 5 5 7)', 0) )
```

Ejemplo 1:

Este ejemplo cambia todas las apariciones del punto (5, 5) por el punto (6, 6) en la cadena lineal.

```
SELECT cast(ST_AsText(ST_ChangePoint(line, ST_Point(5, 5),
                                     ST_Point(6, 6))) as VARCHAR(160))
FROM   sample_lines
WHERE  id=1
```

Resultados:

```
NEW
--------------------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 6.00000000 6.00000000, 0.00000000
0.00000000, 10.00000000 0.00000000, 6.00000000 6.00000000, 0.00000000
10.00000000)
```

Ejemplo 2:

Este ejemplo cambia todas las apariciones del punto (5, 5, 5) por el punto (6, 6, 6) en la cadena lineal.

```
SELECT cast(ST_AsText(ST_ChangePoint(line, ST_Point(5.0, 5.0, 5.0),
       ST_Point(6.0, 6.0, 6.0) )) as VARCHAR(180))
FROM   sample_lines
WHERE  id=2
```

Resultados:

```
NEW
----------------------------------------------------------------------------
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 6.00000000 6.00000000
6.00000000, 10.00000000 10.00000000 6.00000000, 5.00000000 5.00000000
7.00000000)
```

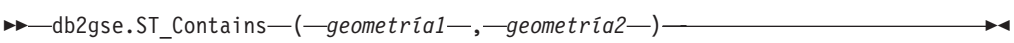
ST_Contains

ST_Contains toma dos geometrías como parámetros de entrada y devuelve un 1 si la primera geometría contiene completamente a la segunda; de lo contrario devuelve un 0 (cero) para indicar que la primera geometría no contiene completamente a la segunda.

Si cualquiera de las geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

Sintaxis:



Parámetro:

geometría1
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la que se comprobar si contiene completamente a *geometría2*.

geometría2
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la que se comprobar si está completamente dentro de *geometría1*.

Restricciones: Para los datos geodésicos, ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

Tipo de retorno:

INTEGER

Ejemplos:

El código siguiente crea y llena estas tablas:

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_points(id SMALLINT, geometry ST_POINT)

CREATE TABLE sample_lines(id SMALLINT, geometry ST_LINESTRING)

CREATE TABLE sample_polygons(id SMALLINT, geometry ST_POLYGON)

INSERT INTO sample_points (id, geometry)
VALUES
    (1, ST_Point(10, 20, 1)),
    (2, ST_Point('point(41 41)', 1))

INSERT INTO sample_lines (id, geometry)
VALUES
    (10, ST_LineString('linestring (1 10, 3 12, 10 10)', 1) ),
    (20, ST_LineString('linestring (50 10, 50 12, 45 10)', 1) )
```

```
INSERT INTO sample_polygons(id, geometry)
VALUES
      (100, ST_Polygon('polygon((0 0, 0 40, 40 40, 40 0, 0 0))',  1) )
```

Ejemplo 1:

El fragmento de código siguiente utiliza la función ST_Contains para determinar qué puntos están contenidos dentro de un polígono determinado.

```
SELECT poly.id AS polygon_id,
       CASE ST_Contains(poly.geometry, pts.geometry)
          WHEN 0 THEN 'does not contain'
          WHEN 1 THEN 'does contain'
       END AS contains,
       pts.id AS point_id
FROM   sample_points pts, sample_polygons poly
```

Resultados:

```
POLYGON_ID CONTAINS         POINT_ID
---------- ---------------- --------
       100 does contain            1
       100 does not contain        2
```

Ejemplo 2:

El fragmento de código siguiente utiliza la función ST_Contains para determinar qué líneas están contenidas dentro de un polígono determinado.

```
SELECT poly.id AS polygon_id,
       CASE ST_Contains(poly.geometry, line.geometry)
          WHEN 0 THEN 'does not contain'
          WHEN 1 THEN 'does contain
       END AS contains,
       line.id AS line_id
FROM   sample_lines line, sample_polygons poly
```

Resultados:

```
POLYGON_ID CONTAINS         LINE_ID
---------- ---------------- -------
       100 does contain          10
       100 does not contain      20
```

Información relacionada:

- "ST_Within" en la página 512

ST_ConvexHull

ST_ConvexHull toma una geometría como parámetro de entrada y devuelve su cáscara convexa.

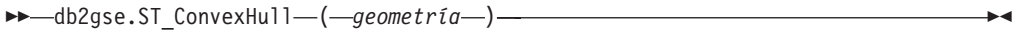
La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si es posible, el tipo específico de la geometría devuelta será ST_Point, ST_LineString o ST_Polygon. Por ejemplo, el perímetro de un polígono sin huecos es una cadena lineal individual, representada como ST_LineString. El perímetro de un polígono con uno o varios huecos consiste en varias cadenas lineales, representadas como ST_MultiLineString.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la cual se debe calcular la cáscara convexa.

Tipo de retorno:

db2gse.ST_Geometry

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente crea y llena la tabla SAMPLE_GEOMETRIES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id INTEGER, spatial_type varchar(18),
   geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, spatial_type, geometry)
VALUES
    (1, 'ST_LineString', ST_LineString
        ('linestring(20 20, 30 30, 20 40, 30 50)', 0)),
    (2, 'ST_Polygon', ST_Polygon('polygon
        ((110  120,  110 140, 120 130, 110 120))', 0) ),
    (3, 'ST_Polygon', ST_Polygon('polygon((30 30, 25 35, 15 50,
        35 80, 40 85, 80 90,70 75, 65 70, 55 50, 75 40, 60 30,
        30 30))', 0) ),
    (4, 'ST_MultiPoint', ST_MultiPoint('multipoint(20 20, 30 30,
        20 40, 30 50)', 1))
```

La sentencia SELECT siguiente calcula la cáscara convexa de todas las geometrías creadas anteriormente y visualiza el resultado.

```
SELECT id, spatial_type, cast(geometry..ST_ConvexHull..ST_AsText
   AS varchar(300)) AS convexhull
FROM   sample_geometries
```

Resultados:

```
ID     SPATIAL_TYPE       CONVEXHULL
------ ------------------ ----------------------------------------------------
    1 ST_LineString      POLYGON (( 20.00000000 40.00000000,
                           20.00000000 20.00000000, 30.00000000
                           30.00000000, 30.00000000 50.00000000,
                           20.00000000 40.00000000))

    2 ST_Polygon         POLYGON (( 110.00000000 140.00000000,
                           110.00000000 120.00000000, 120.00000000
                           130.00000000,  110.00000000 140.00000000))
```

```
3 ST_Polygon         POLYGON (( 15.00000000 50.00000000,
                       25.00000000 35.00000000, 30.00000000
                       30.00000000, 60.00000000 30.00000000,
                       75.00000000 40.00000000, 80.00000000
                       90.00000000, 40.00000000 85.00000000,
                       35.00000000 80.00000000, 15.00000000
                       50.00000000))

4 ST_MultiPoint      POLYGON (( 20.00000000 40.00000000,
                       20.00000000 20.00000000, 30.00000000
                       30.00000000, 30.00000000 50.00000000,
                       20.00000000 40.00000000))
```

ST_CoordDim

ST_CoordDim toma una geometría como parámetro de entrada y devuelve la dimensionalidad de sus coordenadas.

Si la geometría proporcionada no tiene coordenadas Z ni M, la dimensionalidad es 2. Si tiene coordenadas Z y no tiene M coordenadas, o si tiene coordenadas M y no tiene coordenadas Z, la dimensionalidad es 3. Si tiene coordenadas Z y M, la dimensionalidad es 4. Si la geometría es un valor nulo, se vuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría de la que se debe recuperar la dimensionalidad.

Tipo de retorno:

INTEGER

Ejemplo:

Este ejemplo crea varias geometrías y luego determina la dimensionalidad de sus coordenadas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id CHARACTER(15), geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       ('Empty Point', ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
       ('Linestring', ST_Geometry('linestring (10 10, 15 20)',0))

INSERT INTO sample_geoms VALUES
       ('Polygon', ST_Geometry('polygon((40 120, 90 120, 90 150,
         40 150, 40 120))' ,0))

INSERT INTO sample_geoms VALUES
       ('Multipoint M', ST_Geometry('multipoint m (10 10 5, 50 10
         6, 10 30 8)' ,0))
```

```
INSERT INTO sample_geoms VALUES
       ('Multipoint Z', ST_Geometry('multipoint z (47 34 295,
         23 45 678)' ,0))

INSERT INTO sample_geoms VALUES
       ('Point ZM',     ST_Geometry('point zm (10 10 16 30)' ,0))


SELECT id, ST_CoordDim(geometry)  COORDDIM
FROM sample_geoms
```

Resultados:

```
ID          COORDDIM
--------------- -----------
Empty Point               2
Linestring                2
Polygon                   2
Multipoint M              3
Multipoint Z              3
Point ZM                  4
```

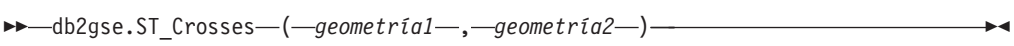
ST_Crosses

ST_Crosses toma dos geometrías como parámetros de entrada y devuelve un 1 si la primera geometría forma intersección con la segunda. En caso contrario, se devuelve un 0 (cero).

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si la primera geometría es un polígono o un multipolígono, o si la segunda geometría es un punto o un multipunto, o bien si cualquiera de la geometrías es un valor nulo, se devuelve un valor nulo. Si la dimensión de la geometría resultante de la intersección de las dos geometrías es una unidad menor que la dimensión máxima de las dos geometrías, y la geometría no es igual a ninguna de esas dos geometrías, se devuelve un 1. En otro caso, el resultado es 0 (cero).

Sintaxis:



Parámetro:

geometría1
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la cual se debe verificar si forma intersección con *geometría2*.

geometría2
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la que se debe comprobar si forma intersección con *geometría1*.

Tipo de retorno:

INTEGER

Ejemplo:

El código siguiente determina si las geometrías creadas forman intersección entre sí.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (1, ST_Geometry('polygon((30 30, 30 50, 50 50, 50 30, 30 30))' ,0))

INSERT INTO sample_geoms VALUES
       (2, ST_Geometry('linestring(40 50, 50 40)' ,0))

INSERT INTO sample_geoms VALUES
       (3, ST_Geometry('linestring(20 20, 60 60)' ,0))

SELECT a.id, b.id, ST_Crosses(a.geometry, b.geometry) Crosses
FROM   sample_geoms a, sample_geoms b
```

Resultados:

```
ID          ID          CROSSES
----------- ----------- -----------
          1           1           -
          2           1           0
          3           1           1
          1           2           -
          2           2           0
          3           2           1
          1           3           -
          2           3           1
          3           3           0
```

Información relacionada:

• "Funciones que comparan elementos geográficos" en la página 316

ST_Difference

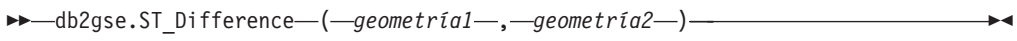
ST_Difference toma dos geometrías como parámetros de entrada y devuelve la parte de la primera geometría que no forma intersección con la segunda geometría.

Ambas geometrías deben tener la misma dimensión. Si cualquiera de las geometrías es un valor nulo, se devuelve un valor nulo. Si la primera dimensión está vacía, se devuelve una dimensión vacía de tipo ST_Point. Si la segunda geometría está vacía, se devuelve la primera geometría sin modificar.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría1
Valor de tipo ST_Geometry que representa la primera geometría que se debe utilizar para calcular la diferencia con *geometría2*.

geometría2
Valor de tipo ST_Geometry que representa la segunda geometría que se utiliza para calcular la diferencia con *geometría1*.

Restricciones para los datos geodésicos:

- Ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.
- ST_Difference sólo soporta los tipos de datos ST_Point, ST_Polygon, ST_MultiPoint y ST_MultiPolygon.

Tipo de retorno:

db2gse.ST_Geometry

La dimensión de la geometría devuelta es la misma que la de las geometrías de entrada.

Ejemplos:

En el ejemplo siguiente, los resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente crea y llena la tabla SAMPLE_GEOMETRIES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (1, ST_Geometry('polygon((10 10, 10 20, 20 20, 20 10, 10 10))' ,0))

INSERT INTO sample_geoms VALUES
       (2, ST_Geometry('polygon((30 30, 30 50, 50 50, 50 30, 30 30))' ,0))

INSERT INTO sample_geoms VALUES
       (3, ST_Geometry('polygon((40 40, 40 60, 60 60, 60 40, 40 40))' ,0))

INSERT INTO sample_geoms VALUES
       (4, ST_Geometry('linestring(70 70, 80 80)' ,0))

INSERT INTO sample_geoms VALUES
       (5, ST_Geometry('linestring(75 75, 90 90)' ,0))
```

Ejemplo 1:

Este ejemplo busca la diferencia entre dos polígonos inconexos.

```
SELECT a.id, b.id, CAST(ST_AsText(ST_Difference(a.geometry, b.geometry))
   as VARCHAR(200))  Difference
FROM sample_geoms a, sample_geoms b
WHERE a.id = 1 and b.id = 2
```

Resultados:

```
ID       ID       DIFFERENCE
-------- -------- --------------------------------------------------
       1        2 POLYGON (( 10.00000000 10.00000000, 20.00000000
                     10.00000000, 20.00000000 20.00000000,
                     10.00000000 20.00000000, 10.00000000 10.00000000))
```

Ejemplo 2:

Este ejemplo busca la diferencia entre dos polígonos que forman una intersección.

```
SELECT a.id, b.id, CAST(ST_AsText(ST_Difference(a.geometry, b.geometry))
   as VARCHAR(200))  Difference
FROM sample_geoms a, sample_geoms b
WHERE a.id = 2 and b.id = 3
```

Resultados:

```
ID       ID       DIFFERENCE
-------- -------- --------------------------------------------------
       2        3 POLYGON (( 30.00000000 30.00000000, 50.00000000
                     30.00000000, 50.00000000 40.00000000, 40.00000000
                     40.00000000, 40.00000000 50.00000000, 30.00000000
                     50.00000000, 30.00000000 30.00000000))
```

Ejemplo 3:

Este ejemplo busca la diferencia entre dos cadenas lineales que se solapan.

```
SELECT a.id, b.id, CAST(ST_AsText(ST_Difference(a.geometry, b.geometry))
   as VARCHAR(100))  Difference
FROM sample_geoms a, sample_geoms b
WHERE a.id = 4 and b.id = 5
```

Resultados:

```
ID       ID       DIFFERENCE
-------- -------- --------------------------------------------------
       4        5 LINESTRING ( 70.00000000 70.00000000, 75.00000000
                     75.00000000)
```

ST_Dimension

ST_Dimension toma una geometría como parámetro de entrada y devuelve la dimensión del mismo.

Si la geometría proporcionada está vacía, se devuelve el valor -1. Para puntos y multipuntos, la dimensión es cero 0 (cero); para curvas y multicurvas, la dimensión es 1; y para polígonos y multipolígonos, la dimensión es 2. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry que representa la geometría para la que se devuelve la dimensión.

Tipo de retorno:

INTEGER

Ejemplo:

Este ejemplo crea varias geometrías diferentes y determina sus dimensiones.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id char(15), geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       ('Empty Point', ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
       ('Point ZM', ST_Geometry('point zm (10 10 16 30)' ,0))

INSERT INTO sample_geoms VALUES
       ('MultiPoint M', ST_Geometry('multipoint m (10 10 5,
         50 10 6, 10 30 8)' ,0))

INSERT INTO sample_geoms VALUES
       ('LineString', ST_Geometry('linestring (10 10, 15 20)',0))

INSERT INTO sample_geoms VALUES
       ('Polygon', ST_Geometry('polygon((40 120, 90 120, 90 150,
         40 150, 40 120))' ,0))

SELECT id, ST_Dimension(geometry)  Dimension
FROM sample_geoms
```

Resultados:

```
ID              DIMENSION
--------------- -----------
Empty Point              -1
Point ZM                  0
MultiPoint M              0
LineString                1
Polygon                   2
```

ST_Disjoint

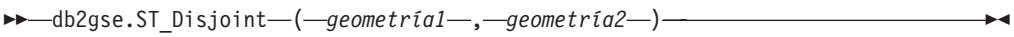
ST_Disjoint toma dos geometrías como parámetros de entrada y devuelve un 1 si dichas geometrías no forman intersección. Si las geometrías forman intersección, se devuelve un 0 (cero).

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si cualquiera de las dos geometrías es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría1
Valor de tipo ST_Geometry que representa la geometría que se prueba si es inconexa respecto a *geometría2*.

geometría2
Valor de tipo ST_Geometry que representa la geometría que se prueba si es inconexa respecto a *geometría1*.

Tipo de retorno:

INTEGER

Ejemplos:

El código siguiente crea varias geometrías en la tabla SAMPLE_GEOMETRIES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (1, ST_Geometry('polygon((20 30, 30 30, 30 40, 20 40, 20 30))',0))

INSERT INTO sample_geoms VALUES
       (2, ST_Geometry('polygon((30 30, 30 50, 50 50, 50 30, 30 30))',0))

INSERT INTO sample_geoms VALUES
       (3, ST_Geometry('polygon((40 40, 40 60, 60 60, 60 40, 40 40))',0))

INSERT INTO sample_geoms VALUES
       (4, ST_Geometry('linestring(60 60, 70 70)' ,0))

INSERT INTO sample_geoms VALUES
       (5, ST_Geometry('linestring(30 30, 40 40)' ,0))
```

Ejemplo 1:

Este ejemplo determina si el primer polígono es inconexo respecto a cualquiera de las geometrías.

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 1
```

Resultados:

```
ID          ID          DISJOINT
----------- ----------- -----------
          1           1           0
          1           2           0
          1           3           1
          1           4           1
          1           5           0
```

Ejemplo 2:

Este ejemplo determina si el tercer polígono es inconexo respecto a cualquiera de las geometrías.

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 3
```

Resultados:

```
ID          ID          DISJOINT
----------- ----------- -----------
          3           1           1
          3           2           0
          3           3           0
          3           4           0
          3           5           0
```

Ejemplo 3:

Este ejemplo determina si la segunda cadena lineal es inconexo respecto a cualquiera de las geometrías.

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 5
```

Resultados:

```
ID          ID          DISJOINT
----------- ----------- -----------
          5           1           0
          5           2           0
          5           3           0
          5           4           1
          5           5           0
```

Información relacionada:

- "Funciones que comparan elementos geográficos" en la página 316

ST_Distance

ST_Distance toma como parámetros de entrada dos geometrías y, opcionalmente, una unidad, y devuelve la distancia más corta entre cualquier punto de la primera geometría y cualquier punto de la segunda geometría, expresada en las unidades proporcionadas o en las unidades por omisión.

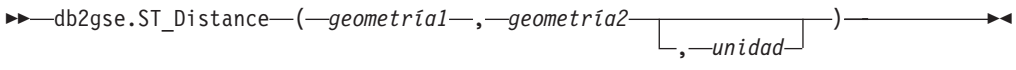
Para los datos geodésicos, ST_Distance devuelve la *distancia geodésica* entre dos geometrías cualquiera. La distancia geodésica es la distancia más corta en la superficie del elipsoide. Para obtener más información, consulte el apartado "Distancias geodésicas" en la página 168.

Si cualquiera de las geometrías es un valor nulo o está vacía, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría1
Valor de tipo ST_Geometry que representa la geometría que se utiliza para calcular la distancia respecto a *geometría2*.

geometría2
Valor de tipo ST_Geometry que representa la geometría que se utiliza para calcular la distancia respecto a *geometría1*.

unidad Valor de tipo VARCHAR(128) que identifica las unidades utilizadas para medir el resultado. Las unidades de medida soportadas están listadas en la vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE.

Para los datos geodésicos, ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

Si se omite el parámetro *unidad*, se siguen las normas siguientes para determinar la unidad de medida utilizada para el resultado:

- Si la *geometría1* está en un sistema de coordenadas proyectadas o geocéntricas, se utiliza la unidad lineal correspondiente a este sistema de coordenadas por omisión.
- Si la *geometría1* está en un sistema de coordenadas geográficas pero no en un SRS geodésico, se utiliza la unidad angular correspondiente a este sistema de coordenadas por omisión.
- Si la *geometría1* está en un SRS geodésico, la unidad de medida por omisión será el metro.

Restricciones para las conversiones de unidades: Se devuelve un error (SQLSTATE 38SU4) si se cumple alguna de las siguientes condiciones:

- La geometría está en un sistema de coordenadas no especificado y se especifica el parámetro *unidad*.
- La geometría está en un sistema de coordenadas proyectadas y se especifica una unidad angular.
- La geometría está en un sistema de coordenadas geográficas pero no en un SRS geodésico y se ha especificado una unidad lineal.
- La geometría está en un sistema de coordenadas geográficas, está en un SRS geodésico y se especifica una unidad angular.

Tipo de retorno:

DOUBLE

Ejemplos:

Las siguientes sentencias de SQL crean y llenan las tablas SAMPLE_GEOMETRIES1 y SAMPLE_GEOMETRIES2.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries1(id SMALLINT, spatial_type varchar(13),
   geometry ST_GEOMETRY)

CREATE TABLE sample_geometries2(id SMALLINT, spatial_type varchar(13),
   geometry ST_GEOMETRY)

INSERT INTO sample_geometries1(id, spatial_type, geometry)
VALUES
    ( 1, 'ST_Point', ST_Point('point(100 100)', 1) ),
    (10, 'ST_LineString', ST_LineString('linestring(125 125, 125 175)', 1) ),
```

```
    (20, 'ST_Polygon', ST_Polygon('polygon
                       ((50 50, 50 150, 150 150, 150 50, 50 50))', 1) )

INSERT INTO sample_geometries2(id, spatial_type, geometry)
VALUES
    (101, 'ST_Point', ST_Point('point(200 200)', 1) ),
    (102, 'ST_Point', ST_Point('point(200 300)', 1) ),
    (103, 'ST_Point', ST_Point('point(200 0)', 1) ),
    (110, 'ST_LineString', ST_LineString('linestring(200 100, 200 200)', 1) ),
    (120, 'ST_Polygon', ST_Polygon('polygon
                        ((200 0, 200 200, 300 200, 300 0, 200 0))', 1) )
```

Ejemplo 1:

La sentencia SELECT siguiente calcula la distancia entre las diversas geometrías contenidas en las tablas SAMPLE_GEOMTRIES1 y SAMPLE_GEOMTRIES2.

```
SELECT   sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
         sg2.id AS sg1_id, sg2.spatial_type AS sg2_type,
         cast(ST_Distance(sg1.geometry, sg2.geometry)
         AS Decimal(8, 4)) AS distance
FROM     sample_geometries1 sg1, sample_geometries2 sg2
ORDER BY sg1.id
```

Resultados:

```
SG1_ID SG1_TYPE      SG1_ID SG2_TYPE      DISTANCE
------ ------------- ------ ------------- ----------
     1 ST_Point         101 ST_Point        141.4213
     1 ST_Point         102 ST_Point        223.6067
     1 ST_Point         103 ST_Point        141.4213
     1 ST_Point         110 ST_LineString   100.0000
     1 ST_Point         120 ST_Polygon      100.0000
    10 ST_LineString    101 ST_Point         79.0569
    10 ST_LineString    102 ST_Point        145.7737
    10 ST_LineString    103 ST_Point        145.7737
    10 ST_LineString    110 ST_LineString    75.0000
    10 ST_LineString    120 ST_Polygon       75.0000
    20 ST_Polygon       101 ST_Point         70.7106
    20 ST_Polygon       102 ST_Point        158.1138
    20 ST_Polygon       103 ST_Point         70.7106
    20 ST_Polygon       110 ST_LineString    50.0000
    20 ST_Polygon       120 ST_Polygon       50.0000
```

Ejemplo 2:

La sentencia SELECT siguiente muestra cómo encontrar todas las geometrías que están dentro de una distancia de 100 la una respecto a las otras.

```
SELECT   sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
         sg2.id AS sg1_id, sg2.spatial_type AS sg2_type,
         cast(ST_Distance(sg1.geometry, sg2.geometry)
         AS Decimal(8, 4)) AS distance
FROM     sample_geometries1 sg1, sample_geometries2 sg2
WHERE    ST_Distance(sg1.geometry, sg2.geometry)  <= 100
```

Resultados:

```
SG1_ID SG1_TYPE      SG1_ID SG2_TYPE      DISTANCE
------ ------------- ------ ------------- ----------
     1 ST_Point         110 ST_LineString   100.0000
     1 ST_Point         120 ST_Polygon      100.0000
    10 ST_LineString    101 ST_Point         79.0569
    10 ST_LineString    110 ST_LineString    75.0000
    10 ST_LineString    120 ST_Polygon       75.0000
```

```
    20 ST_Polygon       101 ST_Point         70.7106
    20 ST_Polygon       103 ST_Point         70.7106
    20 ST_Polygon       110 ST_LineString    50.0000
    20 ST_Polygon       120 ST_Polygon       50.0000
```

Ejemplo 3:

La sentencia SELECT siguiente calcula la distancia en kilómetros entre las diversas geometrías.

```
Tablas SAMPLE_GEOMTRIES1 y SAMPLE_GEOMTRIES2.
SELECT   sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
         sg2.id AS sg1_id, sg2.spatial_type AS sg2_type,
         cast(ST_Distance(sg1.geometry, sg2.geometry, 'KILOMETER')
         AS DECIMAL(10, 4)) AS distance
FROM     sample_geometries1 sg1, sample_geometries2 sg2
ORDER BY sg1.id
```

Resultados:

```
SG1_ID SG1_TYPE      SG1_ID SG2_TYPE      DISTANCE
------ ------------- ------ ------------- ------------
     1 ST_Point         101 ST_Point        12373.2168
     1 ST_Point         102 ST_Point        16311.3816
     1 ST_Point         103 ST_Point         9809.4713
     1 ST_Point         110 ST_LineString    1707.4463
     1 ST_Point         120 ST_Polygon      12373.2168
    10 ST_LineString    101 ST_Point         8648.2333
    10 ST_LineString    102 ST_Point        11317.3934
    10 ST_LineString    103 ST_Point        10959.7313
    10 ST_LineString    110 ST_LineString    3753.5862
    10 ST_LineString    120 ST_Polygon      10891.1254
    20 ST_Polygon       101 ST_Point         7700.5333
    20 ST_Polygon       102 ST_Point        15039.8109
    20 ST_Polygon       103 ST_Point         7284.8552
    20 ST_Polygon       110 ST_LineString    6001.8407
    20 ST_Polygon       120 ST_Polygon      14515.8872
```

Información relacionada:

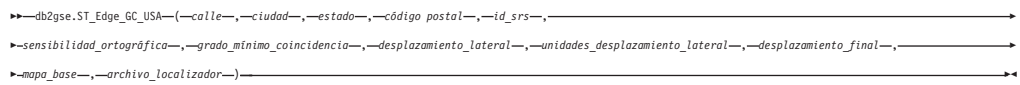
- "Funciones que comparan elementos geográficos" en la página 316

ST_Edge_GC_USA

ST_Edge_GC_USA es la función que hace uso del geocodificador DB2SE_USA_GEOCODER para geocodificar direcciones situadas en Estados Unidos en forma de puntos. Las direcciones se comparan con archivos EDGE, que se proporcionan en el CD de datos del geocodificador.

Esta función toma como parámetros de entrada el nombre y el número de la calle, el nombre de la ciudad, el estado, el código postal y el identificador de sistema de referencia espacial del punto resultante y devuelve un valor de tipo ST_Point. Además, se pueden especificar varios parámetros de configuración que afectan al proceso de geocodificación.

Sintaxis:



Parámetro:

calle Valor de tipo VARCHAR(128) que contiene el número y nombre de la calle correspondientes a la dirección que se debe geocodificar.

Este valor no debe ser nulo.

ciudad Valor de tipo VARCHAR(128) que contiene el nombre de la ciudad correspondiente a la dirección que se debe geocodificar.

Este valor puede ser nulo si se especifica el parámetro *código postal*.

estado Valor de tipo VARCHAR(128) que contiene el nombre del estado correspondiente a la dirección que se debe geocodificar. El estado se puede especificar de forma abreviada o utilizando su nombre completo.

Este valor puede ser nulo si se especifica el parámetro *código postal*.

código postal
Valor de tipo VARCHAR(10) que contiene el código postal de la dirección que se debe geocodificar. El código postal se puede especificar en forma de 5 dígitos o utilizando la notación de 5+4.

Este valor puede ser nulo si se especifican los parámetros *ciudad* y *estado*.

id_srs Valor de tipo INTEGER que contiene el identificador numérico del sistema de referencia espacial correspondiente al punto resultante. El valor debe identificar un sistema de referencia espacial existente, que hace uso de un sistema de coordenadas proyectadas basado en el sistema de coordenadas geográficas GCS_NORTH_AMERICAN_1983, o un sistema de referencia espacial que hace uso del propio sistema de coordenadas geográficas, GCS_NORTH_AMERICAN_1983.

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite un error (SQLSTATE 38SU1).

sensibilidad_ortográfica
Valor de tipo INTEGER que especifica la sensibilidad ortográfica que se debe aplicar a la dirección proporcionada. El valor debe estar comprendido entre 0 (cero) y 100. Cuanto más alto sea este valor, más estricto será el geocodificador respecto a las diferencias en la ortografía de la dirección proporcionada. Las desviaciones detectadas dan como resultado una mayor penalización que se aplicará al grado final de coincidencia.

Si la sensibilidad ortográfica tiene un valor demasiado alto, probablemente se codificarán satisfactoriamente menos direcciones, y en su lugar se devolverá un valor nulo. Si la sensibilidad ortográfica tiene un valor demasiado bajo, pueden ser aceptadas como correctas más direcciones no apropiadas, debido al nivel aceptado para las diferencias ortográficas de las direcciones. **Recomendación:** Asigne el valor 60 a este parámetro.

Si este valor es nulo, la sensibilidad ortográfica se obtendrá a partir del archivo localizador. Si el valor no está especificado en el archivo localizador, se utilizará una sensibilidad ortográfica igual a 60.

grado_mínimo_coincidencia
Valor de tipo INTEGER que contiene el grado mínimo de coincidencia que un punto debe tener para que sea considerado como coincidente con la dirección proporcionada. Este valor debe estar comprendido entre 0 (cero) y 100. Si el grado de coincidencia del punto es menor que *grado_mínimo_coincidencia*, se devuelve un valor nulo en lugar del punto, y la dirección no se geocodifica.

Diversos factores afectan al grado de coincidencia de un punto, tales como la calidad del mapa base, la sensibilidad ortográfica y la exactitud de la dirección. **Recomendación:** Asigne el valor 80 a este parámetro.

Si este valor es nulo, el grado mínimo de coincidencia se obtendrá a partir del archivo localizador. Si el valor no está especificado en el archivo localizador, se utilizará un grado mínimo de coincidencia igual a 80.

desplazamiento_lateral
Valor de tipo DOUBLE que especifica la distancia a la que debe colocarse un punto resultante respecto el centro de la calle. Este valor debe ser mayor o igual que 0 (cero). El parámetro *unidades_desplazamiento_lateral* identifica las unidades utilizadas para medir el desplazamiento lateral.

Si este valor es nulo, el desplazamiento lateral se obtendrá a partir del archivo localizador. Si el valor no está especificado en el archivo localizador, se utilizará un desplazamiento lateral igual a 0.0.

unidades_desplazamiento_lateral
Valor de tipo VARCHAR(128) que contiene las unidades utilizadas ara medir el parámetro *desplazamiento_lateral*. El valor debe ser una de estas unidades:

- Pulgadas
- Puntos
- Pies
- Yardas
- Millas
- Millas náuticas
- Milímetros
- Centímetros
- Metros
- Kilómetros
- Grados decimales
- Metros proyectados
- Unidades de datos de referencia

Si este valor es nulo, las unidades del desplazamiento lateral se obtendrán a partir del archivo localizador. Si el valor no está especificado en el archivo localizador, el desplazamiento lateral se medirá en pies.

desplazamiento_final
Valor de tipo INTEGER que indica la posición en un segmento de un punto que está exactamente al final de un segmento de calle. Este valor debe ser mayor o igual que 0 (cero). Este parámetro se utiliza para evitar colocar los puntos resultantes en el centro de una calle en las intersecciones. El desplazamiento final se mide en puntos (la resolución más alta posible) en el mapa base.

Si este valor es nulo, el desplazamiento final se obtendrá a partir del archivo localizador. Si el valor no está especificado en el archivo localizador, se utilizará un desplazamiento final igual a 3.

mapa_base
Valor de tipo VARCHAR(256) que contiene el nombre base y la vía de acceso totalmente calificada del archivo del mapa base (.edg). El geocodificador utiliza el archivo del mapa base para compararlo con las direcciones proporcionadas. Deben utilizarse los mapas base

proporcionados por DB2 Spatial Extender. Puede utilizar este parámetro si ha colocado los mapas base en un directorio diferente.

Si este valor es nulo, la vía de acceso del mapa base se obtendrá a partir del archivo localizador. Si la vía de acceso no está especificada en el archivo localizador, el mapa base se buscará en el directorio sqllib de la instancia actual, en el subdirectorio gse/refdata. El nombre base del archivo buscado es usa.edg.

archivo_localizador
Valor de tipo VARCHAR(256) que contiene el nombre base y la vía de acceso totalmente calificada del archivo localizador que contiene parámetros de configuración adicionales del geocodificador. Debe utilizarse el archivo localizador proporcionado por DB2 Spatial Extender.

Si este valor es nulo, el archivo localizador se buscará en el directorio sqllib de la instancia actual, en el subdirectorio gse/cfg/geocoder. El nombre base del archivo buscado es EDGELocator.loc.

Tipo de retorno:

db2gse.ST_Point

Ejemplos:

Ejemplo 1:

El código siguiente crea una tabla SAMPLE_GEOCODING e inserta dos direcciones que luego se geocodifican. El grado mínimo de coincidencia se establece en 50 para las direcciones proporcionadas, y el sistema de referencia espacial de los puntos resultantes es 1.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geocoding (
 street VARCHAR(128),
  city   VARCHAR(128),
 state  VARCHAR(128),
 zip    VARCHAR(5) )

INSERT INTO geocoding(street, city, state, zip)
 VALUES ('1212 New York Ave NW', 'Washington', 'DC', '20005'),
 ('100 First North Street', 'San Jose', 'CA', NULL)

SELECT VARCHAR(ST_AsText(ST_Edge_GC_USA(street, city, state, zip, 1,
    CAST(NULL AS INTEGER), 50, CAST(NULL AS DOUBLE),
    CAST(NULL AS VARCHAR(128)), CAST(NULL AS INTEGER),
     CAST(NULL AS VARCHAR(256)), CAST(NULL AS VARCHAR(256)))), 50)
FROM sample_geocoding
```

Resultados:

```
1
-----------------------------------------------
POINT ( -77.02829300 38.90049000)
POINT ( -121.94507200 37.28766700)
```

Ejemplo 2:

Este ejemplo crea un sistema de referencia espacial que hace uso de un sistema de coordenadas proyectadas. Para simplificar la interfaz de la función de geocodificación, se crea una función definida por el usuario para encapsular la función ST_Edge_GC_USA.

```
db2se create_srs <nombre_bd> -srsName CALIFORNIA -srsId 101 -xScale 1
  -coordsysName NAD_1983_STATEPLANE_CALIFORNIA_I_FIPS_0401

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE FUNCTION California_GC (
  street VARCHAR(128), city VARCHAR(128), zip VARCHAR(10))
 RETURNS db2gse.ST_Point
   LANGUAGE SQL
   RETURN db2gse.ST_Edge_GC_USA(street, city, 'CA', zip, 101,
     CAST(NULL AS INTEGER), CAST(NULL AS INTEGER),
     CAST(NULL AS DOUBLE), CAST(NULL AS VARCHAR(128)),
       CAST(NULL AS INTEGER), CAST(NULL AS VARCHAR(256)))

CREATE TABLE sample_geocoding (
  street VARCHAR(128),
  city  VARCHAR(128),
  state  VARCHAR(128),
    zip    VARCHAR(5) )

INSERT INTO geocoding(street, city, state, zip)
VALUES ('100 First North Street', 'San Jose', 'CA', NULL)

SELECT VARCHAR(ST_AsText(California_GC(street, city, zip)), 50)
FROM sample_geocoding
```

Resultados:

```
1
--------------------------------------------------
POINT ( 2004879.00000000 272723.00000000)

NetBIOS
```

Nota: Los valores de las coordenadas X e Y de los puntos son diferentes a los del primer ejemplo debido a que se utiliza un sistema de referencia espacial diferente.

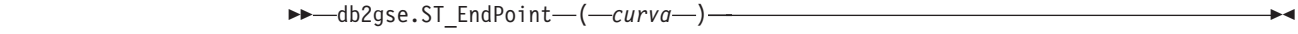
ST_Endpoint

ST_Endpoint toma una curva como parámetro de entrada y devuelve el punto que es el último punto de la curva. El punto resultante se representa según el sistema de referencia espacial de la curva proporcionada.

Si la curva proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

curva Valor de tipo ST_Curve que representa la geometría para la que se devuelve el último punto.

Tipo de retorno:

db2gse.ST_Point

Ejemplo:

La sentencia SELECT siguiente determina el punto final de cada una de las geometrías de la tabla SAMPLE_LINES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines(id INTEGER, line ST_Linestring)


INSERT INTO sample_lines VALUES
    (1, ST_LineString('linestring (10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0) )

INSERT INTO sample_lines VALUES
    (2, ST_LineString('linestring z (0 0 4, 5 5 5, 10 10 6, 5 5 7)', 0) )


SELECT id, CAST(ST_AsText(ST_EndPoint(line)) as VARCHAR(50)) Endpoint
FROM   sample_lines
```

Resultados:

```
ID          ENDPOINT
----------- --------------------------------------------------
          1 POINT ( 0.00000000 10.00000000)
          2 POINT Z ( 5.00000000 5.00000000 7.00000000)
```

Información relacionada:

- "ST_PointN" en la página 483

ST_Envelope

ST_Envelope toma una geometría como parámetro de entrada y devuelve la envoltura de la geometría. La envoltura es un rectángulo que está representado por un polígono.

Si la geometría proporcionada es un punto, una cadena lineal horizontal o una cadena lineal vertical, se devuelve un rectángulo, que es ligeramente mayor que la geometría proporcionada. En otro caso, se devuelve como envoltura el rectángulo delimitador mínimo de la geometría. Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo. Para obtener el rectángulo delimitador mínimo exacto para todas las geometrías, utilice la función ST_MBR.

Para los datos geodésicos, la envoltura es un polígono que delimita el círculo delimitador mínimo de la geometría.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry que representa la geometría para la que se devuelve la envoltura.

Tipo de retorno:

db2gse.ST_Polygon

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo crea varias geometrías y luego determina sus envolturas. Para un punto y una cadena lineal (horizontal) no vacíos, la envoltura es un rectángulo que es ligeramente mayor que la geometría.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
    (1, ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
    (2, ST_Geometry('point zm (10 10 16 30)' ,0))

INSERT INTO sample_geoms VALUES
    (3, ST_Geometry('multipoint m (10 10 5, 50 10 6, 10 30 8)' ,0))

INSERT INTO sample_geoms VALUES
    (4, ST_Geometry('linestring (10 10, 20 10)',0))

INSERT INTO sample_geoms VALUES
    (5, ST_Geometry('polygon((40 120, 90 120, 90 150, 40 150, 40 120))',0))


SELECT id, CAST(ST_AsText(ST_Envelope(geometry)) as VARCHAR(160))  Envelope
FROM sample_geoms
```

Resultados:

```
ID          ENVELOPE
----------- ----------------------------------------------------------------
          1 -

          2 POLYGON (( 9.00000000 9.00000000, 11.00000000 9.00000000,
  11.00000000 11.00000000, 9.00000000 11.00000000, 9.00000000 9.00000000))

          3 POLYGON (( 10.00000000 10.00000000, 50.00000000 10.00000000,
  50.00000000 30.00000000, 10.00000000 30.00000000, 10.00000000
  10.00000000))

          4 POLYGON (( 10.00000000 9.00000000, 20.00000000 9.00000000,
  20.00000000  11.00000000, 10.00000000 11.00000000, 10.00000000
  9.00000000))

          5 POLYGON (( 40.00000000 120.00000000, 90.00000000 120.00000000,
  90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
  120.00000000))
```

Información relacionada:

• "ST_MBR" en la página 442

ST_EnvIntersects

ST_EnvIntersects toma dos geometrías como parámetros de entrada y devuelve un 1 si las envolturas de las dos geometrías forman intersección. En caso contrario, se devuelve un 0 (cero).

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si cualquiera de las geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.

Sintaxis:

Parámetro:

geometría1
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría cuya envoltura se debe verificar si forma intersección con la envoltura de *geometría2*.

geometría2
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría cuya envoltura se debe verificar si forma intersección con la envoltura de *geometría1*.

Tipo de retorno:

INTEGER

Ejemplo:

Este ejemplo crea dos cadenas lineales paralelas y comprueba si forman intersección. Las cadenas lineales propiamente dichas no forman intersección, pero sí lo hacen sus envolturas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (1, ST_Geometry('linestring (10 10, 50 50)',0))

INSERT INTO sample_geoms VALUES
       (2, ST_Geometry('linestring (10 20, 50 60)',0))

SELECT a.id, b.id, ST_Intersects(a.geometry, b.geometry) Intersects,
                   ST_EnvIntersects(a.geometry, b.geometry) Envelope_Intersects
FROM   sample_geoms a , sample_geoms b
WHERE  a.id = 1 and b.id=2
```

Resultados:

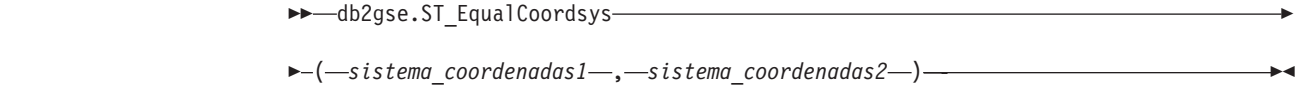
```
ID          ID          INTERSECTS  ENVELOPE_INTERSECTS
----------- ----------- ----------- -------------------
          1           2           0                   1
```

ST_EqualCoordsys

ST_EqualCoordsys toma dos definiciones de sistemas de coordenadas como parámetros de entrada y devuelve el valor entero 1 (uno) si las definiciones proporcionadas son iguales. En otro caso, se devuelve el valor entero 0 (cero). Las definiciones del sistema de coordenadas se comparan independientemente de las diferencias de espacios, paréntesis, caracteres en mayúsculas y minúsculas, y la representación de números de coma flotante.

Si cualquiera de las definiciones de sistemas de coordenadas tiene un valor nulo, se devuelve un valor nulo.

Sintaxis:



Parámetro:

sistema_coordenadas1
: Valor de tipo VARCHAR(2048) que define el primer sistema de coordenadas que se debe comparar con *sistema_coordenadas2*.

sistema_coordenadas2
: Valor de tipo VARCHAR(2048) que define el segundo sistema de coordenadas que se debe comparar con *sistema_coordenadas1*.

Tipo de retorno:

INTEGER

Ejemplo:

Este ejemplo compara dos sistemas de coordenadas australianos para ver si son iguales.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

VALUES ST_EqualCoordSys(
     (SELECT definition
      FROM   db2gse.ST_COORDINATE_SYSTEMS
      WHERE  coordsys_name='GCS_AUSTRALIAN') ,

     (SELECT definition
      FROM   db2gse.ST_COORDINATE_SYSTEMS
      WHERE  coordsys_name='GCS_AUSTRALIAN_1984')
      )
```

Resultados:

```
1
-----------
          0
```

Información relacionada:

- "Vista de catálogo DB2GSE.ST_COORDINATE_SYSTEMS" en la página 295

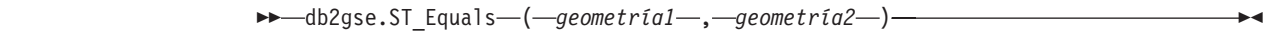
ST_Equals

ST_Equals toma dos geometrías como parámetros de entrada y devuelve un 1 si las geometrías son iguales. En caso contrario, se devuelve un 0 (cero). El orden de los puntos utilizados para definir la geometría no es importante para comprobar la igualdad.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si cualquiera de las dos geometrías proporcionadas es un valor nulo, se devuelve un valor nulo.

Sintaxis:



Parámetro:

geometría1
: Valor de tipo ST_Geometry que representa la geometría que se va a comparar con la *geometría2*.

geometría2
: Valor de tipo ST_Geometry que representa la geometría que se va a comparar con la *geometría1*.

Tipo de retorno:

INTEGER

Ejemplos:

Ejemplo 1:

Este ejemplo crea dos polígonos que tienen sus coordenadas en un orden diferente. ST_Equal muestra que estos polígonos son considerados como iguales.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (1, ST_Geometry('polygon((50 30, 30 30, 30 50, 50 50, 50 30))' ,0))

INSERT INTO sample_geoms VALUES
       (2, ST_Geometry('polygon((50 30, 50 50, 30 50, 30 30, 50 30))' ,0))
```

```
SELECT a.id, b.id, ST_Equals(a.geometry, b.geometry) Equals
FROM   sample_geoms a, sample_geoms b
WHERE a.id = 1 and b.id = 2
```

Resultados:

```
ID          ID          EQUALS
----------- ----------- -----------
          1           2           1
```

Ejemplo 2:

Este ejemplo crea dos geometrías con las mismas coordenadas X e Y, pero con coordenadas M (medidas) diferentes. Cuando las geometrías se comparan mediante la función ST_Equal, se devuelve un 0 (cero) para indicar que las geometrías no son iguales.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (3, ST_Geometry('multipoint m(80 80 6, 90 90 7)' ,0))

INSERT INTO sample_geoms VALUES
       (4, ST_Geometry('multipoint m(80 80 6, 90 90 4)' ,0))


SELECT a.id, b.id, ST_Equals(a.geometry, b.geometry) Equals
FROM   sample_geoms a, sample_geoms b
WHERE a.id = 3 and b.id = 4
```

Resultados:

```
ID          ID          EQUALS
----------- ----------- -----------
          3           4           0
```

Ejemplo 3:

Este ejemplo crea dos geometrías que tienen un conjunto diferente de coordenadas, pero que representan la misma geometría. ST_Equal compara las geometrías e indica que son ciertamente iguales.

```
SET current function path = current function path, db2gse
CREATE TABLE sample_geoms ( id INTEGER, geometry ST_Geometry )

INSERT INTO sample_geoms VALUES
   (5, ST_LineString('linestring ( 10 10, 40 40 )', 0)),
   (6, ST_LineString('linestring ( 10 10, 20 20, 40 40)', 0))

SELECT a.id, b.id, ST_Equals(a.geometry, b.geometry) Equals
FROM   sample_geoms a, sample_geoms b
WHERE  a.id = 5 AND b.id = 6
```

Resultados:

```
ID          ID          EQUALS
----------- ----------- -----------
          5           6           1
```

Información relacionada:

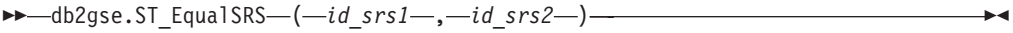
• "Funciones que comparan elementos geográficos" en la página 316

ST_EqualSRS

ST_EqualSRS toma dos identificadores de sistemas de referencia espacial como parámetros de entrada y devuelve un 1 si los sistemas de referencia espacial son iguales. En caso contrario, se devuelve un 0 (cero). Se comparan los desplazamientos, factores de escala y sistemas de coordenadas.

Si cualquiera de los identificadores de sistemas de coordenadas tiene un valor nulo, se devuelve un valor nulo.

Sintaxis:



Parámetro:

id_srs1 Valor de tipo INTEGER que identifica el primer sistema de referencia espacial que se debe comparar con el sistema de referencia espacial identificado por *id_srs2*.

id_srs2 Valor de tipo INTEGER que identifica el segundo sistema de referencia espacial que se comparará con el sistema de referencia espacial identificado por *id_srs1*.

Tipo de retorno:

INTEGER

Ejemplo:

Las siguientes llamadas a db2se crean dos sistemas de referencia espacial similares.

```
db2se create_srs SAMP_DB  -srsId 12 -srsName NYE_12 -xOffset 0 -yOffset 0
   -xScale 1 -yScale 1 -coordsysName
   NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet

db2se create_srs SAMP_DB  -srsId 22 -srsName NYE_22 -xOffset 0 -yOffset 0
   -xScale 1 -yScale 1 -coordsysName
   NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet
```

Estos sistemas de referencia espacial tienen los mismos valores de desplazamiento y escala, y hacen referencia a los mismos sistemas de coordenadas. La única diferencia está en el nombre definido y el ID de sistema de referencia espacial. Por tanto, la función de comparación devuelve un 1, que indica que los sistemas de referencia espacial son iguales.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

VALUES ST_EqualSRS(12, 22)
```

Resultados:

```
          1
-----------
          1
```

Información relacionada:

- "Vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS" en la página 303

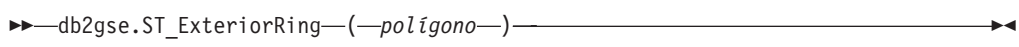
ST_ExteriorRing

ST_ExteriorRing toma un polígono como parámetro de entrada y devuelve su anillo exterior en forma de curva. La curva resultante se representa según el sistema de referencia espacial del polígono proporcionado.

Si el polígono proporcionado es un valor nulo o está vacío, se devuelve un valor nulo. Si el polígono no tiene anillos interiores, el anillo exterior devuelto es igual al perímetro del polígono.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

polígono
Valor de tipo ST_Polygon que representa el polígono para el que se debe devolver el anillo exterior.

Tipo de retorno:

db2gse.ST_Curve

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo crea dos polígonos: uno con dos anillos interiores y otro sin anillos interiores, y luego determina sus anillos exteriores.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys VALUES
       (1, ST_Polygon('polygon((40 120, 90 120, 90 150, 40 150, 40 120),
                      (50 130, 60 130, 60 140, 50 140, 50 130),
                      (70 130, 80 130, 80 140, 70 140, 70 130))' ,0))

INSERT INTO sample_polys VALUES
       (2, ST_Polygon('polygon((10 10, 50 10, 10 30, 10 10))' ,0))


SELECT id, CAST(ST_AsText(ST_ExteriorRing(geometry))
   AS VARCHAR(180)) Exterior_Ring
FROM sample_polys
```

Resultados:

```
ID          EXTERIOR_RING
----------- ----------------------------------------------------------------
          1 LINESTRING ( 40.00000000 120.00000000, 90.00000000
   120.00000000,  90.00000000 150.00000000, 40.00000000 150.00000000,
```

```
40.00000000 120.00000000)

       2 LINESTRING ( 10.00000000 10.00000000, 50.00000000
10.00000000, 10.00000000 30.00000000, 10.00000000 10.00000000)
```

Información relacionada:

• "ST_Boundary" en la página 364

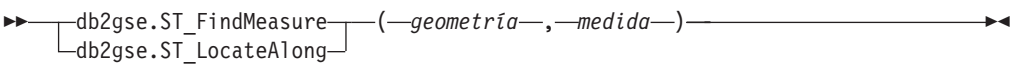
ST_FindMeasure o ST_LocateAlong

ST_FindMeasure o ST_LocateAlong toma una geometría y una medida como parámetros de entrada y devuelve un multipunto o una multicurva que forma parte de la geometría proporcionada y cuya medida es exactamente la especificada. Para los puntos y multipuntos, el resultado es todos los puntos que tienen la medida especificada. En el caso de curvas, multicurvas, superficies y multisuperficies, se realiza una interpolación para calcular el resultado. Para las superficies y multisuperficies, el cálculo se realiza sobre el perímetro de la geometría.

Para los puntos y multipuntos, si no se encuentra la medida proporcionada, el resultado es una geometría vacía. Para todas las demás geometrías, si la medida proporcionada es menor que la medida más pequeña de la geometría o mayor que medida más alta de la geometría, el resultado es una geometría vacía. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría en la que se buscarán partes cuyas coordenadas M (medidas) contengan a *medida*.

medida
: Valor de tipo DOUBLE que es la medida de las partes de *geometría* que se deben incluir en el resultado.

Tipo de retorno:

db2gse.ST_Geometry

Ejemplos:

La sentencia CREATE TABLE siguiente crea la tabla SAMPLE_GEOMETRIES. SAMPLE_GEOMETRIES tiene dos columnas: la columna ID, que sirve para identificar cada columna de forma unívoca, y la columna GEOMETRY de tipo ST_Geometry, que contiene la geometría de ejemplo.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id SMALLINT, geometry ST_GEOMETRY)
```

Las sentencias INSERT siguientes insertan dos filas. La primera es una cadena lineal; la segunda es un multipunto.

```
INSERT INTO sample_geometries(id, geometry)
VALUES
  (1, ST_LineString('linestring m (2 2 3, 3 5 3, 3 3 6, 4 4 8)', 1)),
  (2, ST_MultiPoint('multipoint m
      (2 2 3, 3 5 3, 3 3 6, 4 4 6, 5 5 6, 6 6 8)', 1))
```

Ejemplo 1:

En la sentencia SELECT siguiente y en el resultado correspondiente, la función ST_FindMeasure busca puntos cuya medida es 7. El resultado obtenido para la primera fila es un punto. En cambio, el resultado obtenido para la segunda fila es un punto vacío. En el caso de elementos geográficos lineales (geometrías con una dimensión mayor que 0), ST_FindMeasure puede interpolar el punto; en cambio, para multipuntos, la medida resultante debe coincidir exactamente.

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 7))
   AS varchar(45)) AS measure_7
FROM   sample_geometries
```

Resultados:

```
ID     MEASURE_7
------ ---------------------------------------------
     1 POINT M ( 3.50000000 3.50000000 7.00000000)
     2 POINT EMPTY
```

Ejemplo 2:

En la sentencia SELECT siguiente y en el resultado correspondiente, la función ST_FindMeasure devuelve un punto y un multipunto. La medida resultante (6) coincide con las medidas existentes en las fuentes de datos de ST_FindMeasure y del multipunto.

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 6))
   AS varchar(120)) AS measure_6
FROM   sample_geometries
```

Resultados:

```
ID     MEASURE_6
------ ------------------------------------------------------------
1      POINT M ( 3.00000000 3.00000000 6.00000000)

2      MULTIPOINT M ( 3.00000000 3.00000000 6.00000000, 4.00000000
        4.00000000 6.00000000, 5.00000000 5.00000000 6.00000000)
```

Información relacionada:

- "ST_MeasureBetween, ST_LocateBetween" en la página 445

ST_Generalize

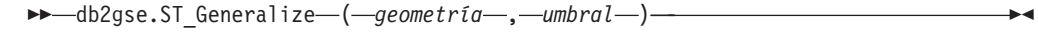
ST_Generalize toma una geometría y un umbral como parámetros de entrada y representa dicha geometría con un número reducido de puntos, al mismo tiempo que conserva las características generales de la geometría. Se utiliza el algoritmo de simplificación de líneas de Douglas-Peucker, mediante el cual la secuencia de puntos que define la geometría se subdivide repetidamente hasta que un tramo de los puntos se puede sustituir por un segmento lineal recto. En este segmento lineal, ninguno de los puntos que lo definen se aparta del segmento lineal recto más allá

del umbral especificado. Las coordenadas Z y M no se tienen en cuenta para la simplificación. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si la geometría proporcionada está vacía, se devuelve una geometría vacía de tipo ST_Point. Si la geometría proporcionada o el umbral son un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la que se aplica la simplificación de línea.

umbral Valor de tipo DOUBLE que identifica el umbral que se utilizará para el algoritmo de simplificación de línea. El umbral debe ser mayor o igual que 0 (cero). Cuanto mayor sea el umbral, menor será el número de puntos que se utilizarán para representar la geometría generalizada. Para los datos geodésicos, la unidad utilizada para el umbral es el metro.

Tipo de retorno:

db2gse.ST_Geometry

Ejemplos:

En los ejemplos siguientes, los resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Se crea una cadena lineal con ocho puntos que van desde (10, 10) hasta (80, 80). El recorrido es casi una línea recta, pero algunos de los puntos están ligeramente fuera de la línea. Se puede utilizar la función ST_Generalize para reducir el número de puntos de la línea.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, geometry ST_LineString)

INSERT INTO sample_lines VALUES
       (1, ST_Linestring('linestring(10 10, 21 20, 34 26, 40 40,
                                     52 50, 59 63, 70 71, 80 80)' ,0))
```

Ejemplo 1:

Cuando se utiliza un factor de generalización igual a 3, la cadena lineal queda reducida a cuatro coordenadas, y sigue siendo muy parecida a la representación original de la cadena lineal.

```
SELECT CAST(ST_AsText(ST_Generalize(geometry, 3)) as VARCHAR(115))
  Generalize_3
FROM sample_lines
```

Resultados:

```
GENERALIZE 3
------------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 34.00000000 26.00000000,
  59.00000000  63.00000000, 80.00000000 80.00000000)
```

Ejemplo 2:

Cuando se utiliza un factor de generalización igual a 6, la cadena lineal queda reducida a sólo dos coordenadas. Esto produce una cadena lineal más simple que el ejemplo anterior, pero se aparta más de la representación original.

```
SELECT CAST(ST_AsText(ST_Generalize(geometry, 6)) as VARCHAR(65))
  Generalize_6
FROM sample_lines
```

Resultados:

```
GENERALIZE 6
------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 80.00000000 80.00000000)
```

ST_GeomCollection

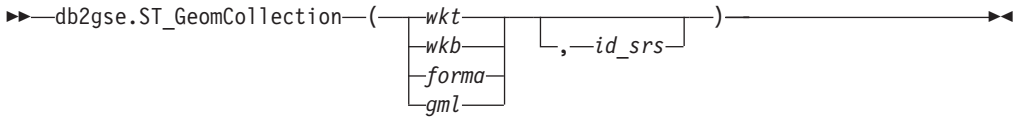
ST_GeomCollection crea una colección de geometrías a partir de uno de estos datos de entrada siguientes:

- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma ESRI
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenida la colección de geometrías resultante.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma ESRI o la representación GML es nula, se devuelve un valor nulo.

Sintaxis:



Parámetro:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la colección de geometrías resultante.

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la colección de geometrías resultante.

forma Valor de tipo BLOB(2G) que contiene la representación de forma ESRI de la colección de geometrías resultante.

gml Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), de la colección de geometrías resultante .

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la colección de geometrías resultante.

Si se omite el parámetro *id_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite un error (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_GeomCollection

Notas:

Si se omite el parámetro *id_srs*, puede ser necesario convertir explícitamente *wkt* y *gml* al tipo de datos CLOB. De lo contrario, DB2 podría recurrir a la función utilizada para convertir valores desde el tipo de referencia REF(ST_GeomCollection) al tipo ST_GeomCollection. El ejemplo siguiente asegura que DB2 recurra a la función correcta:

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST_GeomCollection para crear e insertar un multipunto, una multilínea y un multipolígono de una representación de texto convencional (WKT) y un multipunto de GML (Geographic Markup Language) en una columna GeomCollection.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geomcollections(id INTEGER,
   geometry ST_GEOMCOLLECTION)

INSERT INTO sample_geomcollections(id, geometry)
VALUES
    (4001, ST_GeomCollection('multipoint(1 2, 4 3, 5 6)', 1) ),
    (4002, ST_GeomCollection('multilinestring(
                              (33 2, 34 3, 35 6),
                              (28 4, 29 5, 31 8, 43 12),
                              (39 3, 37 4, 36 7))', 1) ),
    (4003, ST_GeomCollection('multipolygon(((3 3, 4 6, 5 3, 3 3),
                              (8 24, 9 25, 1 28, 8 24),
                              (13 33, 7 36, 1 40, 10 43, 13 33)))', 1)),
    (4004, ST_GeomCollection('<gml:MultiPoint srsName="EPSG:4269"
                             ><gml:PointMember><gml:Point>
                             <gml:coord><gml:X>10</gml:X>
                             <gml:Y>20</gml:Y></gml: coord></gml:Point>
                             </gml:PointMember><gml:PointMember>
                             <gml:Point><gml:coord><gml:X>30</gml:X>
                             <gml:Y>40</gml:Y></gml:coord></gml:Point>
                             </gml:PointMember></gml:MultiPoint>', 1))

SELECT id, cast(geometry..ST_AsText AS varchar(350)) AS geomcollection
FROM   sample_geomcollections
```

Resultados:

```
ID          GEOMCOLLECTION
----------- ------------------------------------------------------------------
4001         MULTIPOINT ( 1.00000000 2.00000000, 4.00000000 3.00000000,
    5.00000000 6.00000000)

4002         MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000
   3.00000000, 35.00000000 6.00000000),( 28.00000000 4.00000000,
   29.00000000 5.00000000, 31.00000000 8.00000000, 43.00000000
   12.00000000),(39.00000000 3.00000000, 37.00000000 4.00000000,
   36.00000000 7.00000000))

4003         MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000
    43.00000000, 1.00000000 40.00000000, 7.00000000 36.00000000,
    13.00000000 33.00000000)),(( 8.00000000 24.00000000, 9.00000000
    25.00000000, 1.00000000 28.00000000, 8.00000000 24.00000000)),
    (( 3.00000000 3.00000000,5.00000000 3.00000000, 4.00000000
    6.00000000,3.00000000 3.00000000)))

4004         MULTIPOINT ( 10.00000000 20.00000000, 30.00000000
    40.00000000)
```

Información relacionada:

- “Representación de texto convencional (WKT)” en la página 529
- “Representación binaria convencional (WKB)” en la página 534
- “Representación de forma” en la página 536
- “Representación GML (Geography Markup Language)” en la página 536

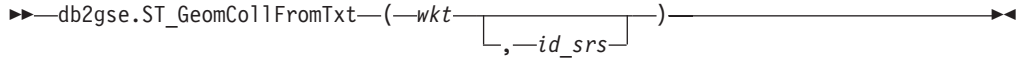
ST_GeomCollFromTxt

ST_GeomCollFromTxt toma como parámetros de entrada una representación de texto convencional de una colección de geometrías y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la colección de geometrías correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST_GeomCollection. Es recomendable debido a su flexibilidad: ST_GeomCollection acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

Sintaxis:



Parámetro:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la colección de geometrías resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la colección de geometrías resultante.

Si se omite el parámetro *id_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite un error (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_GeomCollection

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST_GeomCollFromTxt para crear e insertar un multipunto, una multilínea y un multipolígono de una representación de texto convencional (WKT) en una columna GeomCollection.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geomcollections(id INTEGER, geometry ST_GEOMCOLLECTION)

INSERT INTO sample_geomcollections(id, geometry)
VALUES
    (4011, ST_GeomCollFromTxt('multipoint(1 2, 4 3, 5 6)', 1) ),
    (4012, ST_GeomCollFromTxt('multilinestring(
                           (33 2, 34 3, 35 6),
                           (28 4, 29 5, 31 8, 43 12),
                           (39 3, 37 4, 36 7))', 1) ),
    (4013, ST_GeomCollFromTxt('multipolygon(((3 3, 4 6, 5 3, 3 3),
                           (8 24, 9 25, 1 28, 8 24),
                           (13 33, 7 36, 1 40, 10 43, 13 33)))', 1))

SELECT id, cast(geometry..ST_AsText AS varchar(340))
   AS geomcollection
FROM   sample_geomcollections
```

Resultados:

```
ID          GEOMCOLLECTION
----------- ------------------------------------------------------------------
4011        MULTIPOINT ( 1.00000000 2.00000000, 4.00000000 3.00000000,
   5.00000000 6.00000000)

4012        MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000
   3.00000000, 35.00000000 6.00000000),( 28.00000000 4.00000000, 29.00000000
   5.00000000, 31.00000000 8.00000000, 43.00000000 12.00000000),( 39.00000000
   3.00000000, 37.00000000 4.00000000, 36.00000000 7.00000000))

4013        MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000,
   1.00000000 40.00000000, 7.00000000 36.00000000, 13.00000000 33.00000000)),
   (( 8.00000000 24.00000000, 9.00000000 25.00000000, 1.00000000 28.00000000,
   8.00000000 24.00000000)),(( 3.00000000 3.00000000, 5.00000000 3.00000000,
   4.00000000 6.00000000, 3.00000000 3.00000000)))
```

Información relacionada:

- "ST_GeomCollection" en la página 401

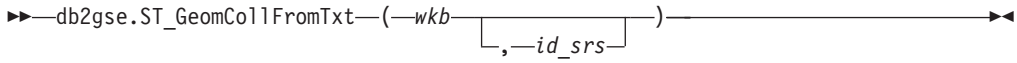
ST_GeomCollFromWKB

ST_GeomCollFromWKB toma como parámetros de entrada una representación binaria convencional de una colección de geometrías y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la colección de geometrías correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La versión preferida para esta función es ST_GeomCollection.

Sintaxis:



Parámetro:

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la colección de geometrías resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la colección de geometrías resultante.

Si se omite el parámetro *id_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite un error (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_GeomCollection

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST_GeomCollFromWKB para crear y consultar las coordenadas de una colección de geometrías en una representación binaria convencional. Las filas se insertan en la tabla SAMPLE_GEOMCOLLECTION con los ID 4021 y 4022, y las colecciones de geometrías en el sistema de referencia espacial 1.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geomcollections(id INTEGER,
  geometry ST_GEOMCOLLECTION, wkb BLOB(32k))

INSERT INTO sample_geomcollections(id, geometry)
VALUES
  (4021, ST_GeomCollFromTxt('multipoint(1 2, 4 3, 5 6)', 1)),
  (4022, ST_GeomCollFromTxt('multilinestring(
                            (33 2, 34 3, 35 6),
                            (28 4, 29 5, 31 8, 43 12))', 1))
```

```
UPDATE sample_geomcollections AS temp_correlated
SET    wkb = geometry..ST_AsBinary
WHERE  id = temp_correlated.id

SELECT id, cast(ST_GeomCollFromWKB(wkb)..ST_AsText
   AS varchar(190)) AS GeomCollection
FROM   sample_geomcollections
```

Resultados:

```
ID          GEOMCOLLECTION
----------- ----------------------------------------------------
       4021 MULTIPOINT ( 1.00000000 2.00000000, 4.00000000
  3.00000000, 5.00000000 6.00000000)

       4022 MULTILINESTRING (( 33.00000000 2.00000000,
 34.00000000 3.00000000, 35.00000000 6.00000000),( 28.00000000
 4.00000000, 29.00000000 5.00000000, 31.00000000 8.00000000,
 43.00000000 12.00000000))
```

Información relacionada:

• "Representación binaria convencional (WKB)" en la página 534

ST_Geometry

ST_Geometry crea una geometría a partir de uno de estos datos de entrada siguientes:

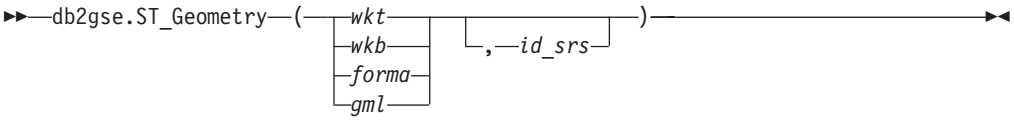
- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma ESRI
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenida la geometría resultante.

El tipo dinámico de la geometría resultante es uno de los subtipos de ST_Geometry de los que pueden crearse instancias.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma ESRI o la representación GML es nula, se devuelve un valor nulo.

Sintaxis:



Parámetro:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la geometría resultante.

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la geometría resultante.

forma Valor de tipo BLOB(2G) que contiene la representación de forma ESRI de la geometría resultante.

gml Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), de la geometría resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la geometría resultante.

Si se omite el parámetro *id_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite un error (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_Geometry

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST_Geometry para crear e insertar un punto desde una representación de texto convencional de un punto o una línea desde una representación GML (Geographic Markup Language) de una línea.

La función ST_Geometry es la más flexible de las funciones constructoras de tipos espaciales, pues puede crear cualquier tipo espacial a partir de diversas representaciones de geometrías. ST_LineFromText sólo puede crear una línea a partir de una representación WKT de una línea. ST_WKTToSql puede crear cualquier tipo, pero sólo a partir de una representación WKT.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id INTEGER, geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, geometry)
VALUES
    (7001, ST_Geometry('point(1 2)', 1) ),
    (7002, ST_Geometry('linestring(33 2, 34 3, 35 6)', 1) ),
    (7003, ST_Geometry('polygon((3 3, 4 6, 5 3, 3 3))', 1)),
    (7004, ST_Geometry('<gml:Point srsName=";EPSG:4269";><gml:coord>
         <gml:X>50</gml:X><gml:Y>60</gml:Y></gml:coord>
           </gml:Point>', 1))

SELECT id, cast(geometry..ST_AsText AS varchar(120)) AS geometry
FROM   sample_geometries
```

Resultados:

```
ID          GEOMETRY
----------- ------------------------------------------------------------
7001        POINT ( 1.00000000 2.00000000)

7002        LINESTRING ( 33.00000000 2.00000000, 34.00000000 3.00000000,
  35.00000000 6.00000000)

7003        POLYGON (( 3.00000000 3.00000000, 5.00000000 3.00000000,
  4.00000000 6.00000000, 3.00000000 3.00000000))

7004        POINT ( 50.00000000 60.00000000)
```

Información relacionada:

- "Representación de texto convencional (WKT)" en la página 529

ST_GeometryN

ST_GeometryN toma una colección de geometrías y un índice como parámetros de entrada y devuelve la geometría de la colección que está identificada por el índice. La geometría resultante se representa según el sistema de referencia espacial de la colección de geometrías proporcionada.

Si la colección de geometrías proporcionada tiene un valor nulo o está vacío, o si el índice es menor que 1 o mayor que el número de geometrías de la colección, se devuelve un valor nulo y se emite una condición de aviso (01HS0).

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

colección
: Valor de tipo ST_GeomCollection o uno de sus subtipos que representa la colección de geometrías dentro de la cual se debe localizar la geometría que ocupa la posición *n*.

índice
: Valor de tipo INTEGER que identifica la geometría que ocupa la posición *n* y que se debe obtener a partir de *colección*.

  Si *índice* es menor que 1 o mayor que el número de geometrías de la colección, se devuelve un valor nulo y se emite un aviso (SQLSTATE 01HS0).

Tipo de retorno:

db2gse.ST_Geometry

Ejemplo:

El código siguiente muestra cómo seleccionar la segunda geometría dentro de la colección de geometrías.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geomcollections (id INTEGER,
  geometry ST_GEOMCOLLECTION)

INSERT INTO sample_geomcollections(id, geometry)
VALUES
    (4001, ST_GeomCollection('multipoint(1 2, 4 3)', 1) ),
    (4002, ST_GeomCollection('multilinestring(
                           (33 2, 34 3, 35 6),
                           (28 4, 29 5, 31 8, 43 12),
                           (39 3, 37 4, 36 7))', 1) ),
    (4003, ST_GeomCollection('multipolygon(((3 3, 4 6, 5 3, 3 3),
                              (8 24, 9 25, 1 28, 8 24),
                              (13 33, 7 36, 1 40, 10 43, 13 33)))', 1))
```

```
SELECT id, cast(ST_GeometryN(geometry, 2)..ST_AsText AS varchar(110))
    AS second_geometry
FROM   sample_geomcollections
```

Resultados:

```
ID          SECOND_GEOMETRY
----------- ------------------------------------------------------------
       4001 POINT ( 4.00000000 3.00000000)

       4002 LINESTRING ( 28.00000000 4.00000000, 29.00000000 5.00000000,
   31.00000000 8.00000000, 43.00000000 12.00000000)

       4003 POLYGON (( 8.00000000 24.00000000, 9.00000000 25.00000000,
   1.00000000 28.00000000, 8.00000000 24.00000000))
```

Información relacionada:

• "ST_NumGeometries" en la página 467

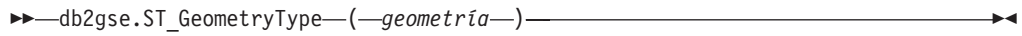
ST_GeometryType

ST_GeometryType toma una geometría como parámetro de entrada y devuelve el nombre de tipo calificado al completo del tipo dinámico de dicha geometría.

Las funciones TYPE_SCHEMA y TYPE_NAME de DB2 tienen el mismo efecto.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry para el que se devolverá el tipo de geometría.

Tipo de retorno:

VARCHAR(128)

Ejemplos:

El código siguiente muestra cómo determinar el tipo de una geometría.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries (id INTEGER, geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, geometry)
VALUES
    (7101, ST_Geometry('point(1 2)', 1) ),
    (7102, ST_Geometry('linestring(33 2, 34 3, 35 6)', 1) ),
    (7103, ST_Geometry('polygon((3 3, 4 6, 5 3, 3 3))', 1)),
    (7104, ST_Geometry('multipoint(1 2, 4 3)', 1) )

SELECT id, geometry..ST_GeometryType AS geometry_type
FROM   sample_geometries
```

Resultados:

```
ID          GEOMETRY_TYPE
----------- ------------------------------
       7101 "DB2GSE  "."ST_POINT"
       7102 "DB2GSE  "."ST_LINESTRING"
       7103 "DB2GSE  "."ST_POLYGON"
       7104 "DB2GSE  "."ST_MULTIPOINT"
```

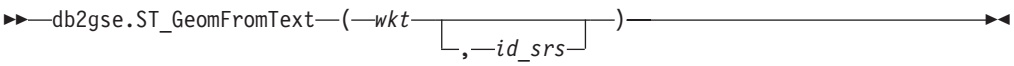
ST_GeomFromText

ST_GeomFromText toma como parámetros de entrada una representación de texto convencional de una geometría y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la geometría correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La versión preferida para esta función es ST_Geometry.

Sintaxis:



Parámetro:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la geometría resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la geometría resultante.

Si se omite el parámetro *id_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite un error (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_Geometry

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

En este ejemplo, se utiliza la función ST_GeomFromText para crear e insertar un punto a partir de su representación de texto convencional (WKT).

El código siguiente inserta filas en la tabla SAMPLE_POINTS con identificadores y geometrías en el sistema de referencia espacial 1 utilizando la representación WKT.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id INTEGER, geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, geometry)
```

```
VALUES
    (1251, ST_GeomFromText('point(1 2)', 1) ),
    (1252, ST_GeomFromText('linestring(33 2, 34 3, 35 6)', 1) ),
    (1253, ST_GeomFromText('polygon((3 3, 4 6, 5 3, 3 3))', 1))
```

La sentencia SELECT siguiente obtiene el ID y las geometrías a partir de la tabla SAMPLE_GEOMETRIES.

```
SELECT id, cast(geometry..ST_AsText AS varchar(105))
   AS geometry
FROM   sample_geometries
```

Resultados:

```
ID          GEOMETRY
----------- ------------------------------------------------------------
       1251 POINT ( 1.00000000 2.00000000)

       1252 LINESTRING ( 33.00000000 2.00000000, 34.00000000 3.00000000,
              35.00000000 6.00000000)

       1253 POLYGON (( 3.00000000 3.00000000, 5.00000000 3.00000000,
              4.00000000 6.00000000, 3.00000000 3.00000000))
```

Información relacionada:

• "Representación de texto convencional (WKT)" en la página 529

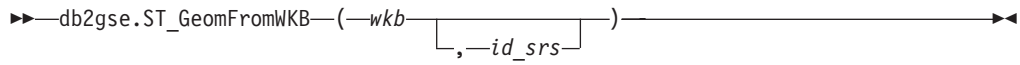
ST_GeomFromWKB

ST_GeomFromWKB toma como parámetros de entrada una representación de texto convencional de una geometría y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la geometría correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La versión preferida para esta función es ST_Geometry.

Sintaxis:



Parámetro:

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la geometría resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la geometría resultante.

Si se omite el parámetro *id_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si el parámetro id_srs no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite un error (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_Geometry

Ejemplos:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST_GeomFromWKB para crear e insertar una línea a partir de una representación binaria convencional (WKB) de una línea.

El ejemplo siguiente inserta un registro en la tabla SAMPLE_GEOMETRIES, con un ID y una geometría en el sistema de referencia espacial 1, en una representación WKB.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries (id INTEGER, geometry ST_GEOMETRY,
    wkb BLOB(32K))

INSERT INTO sample_geometries(id, geometry)
VALUES
    (1901, ST_GeomFromText('point(1 2)', 1) ),
    (1902, ST_GeomFromText('linestring(33 2, 34 3, 35 6)', 1) ),
    (1903, ST_GeomFromText('polygon((3 3, 4 6, 5 3, 3 3))', 1))

UPDATE sample_geometries AS temp_correlated
SET    wkb = geometry..ST_AsBinary
WHERE  id = temp_correlated.id

SELECT id, cast(ST_GeomFromWKB(wkb)..ST_AsText AS varchar(190))
   AS geometry
FROM   sample_geometries
```

Resultados:

```
 ID          GEOMETRY
 ----------- ------------------------------------------------------------
1901         POINT ( 1.00000000 2.00000000)

1902         LINESTRING ( 33.00000000 2.00000000, 34.00000000
   3.00000000, 35.00000000 6.00000000)

1903         POLYGON (( 3.00000000 3.00000000, 5.00000000 3.00000000,
   4.00000000 6.00000000, 3.00000000 3.00000000))
```

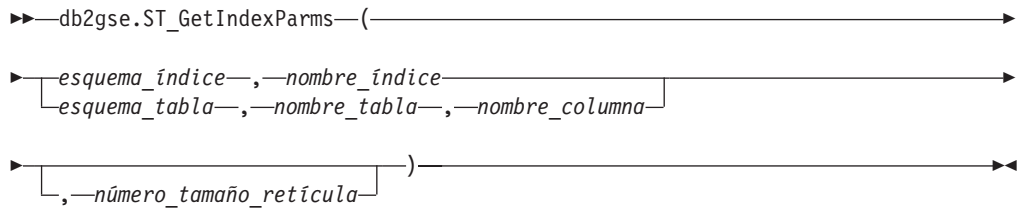
Información relacionada:

• “Representación binaria convencional (WKB)” en la página 534

ST_GetIndexParms

ST_GetIndexParms toma como parámetro de entrada el identificador de un índice espacial o columna espacial y devuelve los parámetros utilizados para definir el índice en la columna espacial. Opcionalmente se puede especificar un número de parámetro, en cuyo caso sólo se obtiene el tamaño de retícula identificado por el número.

Sintaxis:



Parámetro:

esquema_índice
Valor de tipo VARCHAR(128) que identifica el esquema donde está contenido el índice espacial cuyo nombre no calificado es *nombre_índice*. El nombre de esquema distingue entre mayúsculas y minúsculas, y debe aparecer listado en la vista de catálogo SYSCAT.SCHEMATA.

Si este parámetro tiene un valor nulo, se utiliza el valor del registro especial CURRENT SCHEMA como nombre de esquema para el índice espacial.

nombre_índice
Valor de tipo VARCHAR(128) que contiene el nombre no calificado del índice espacial para el cual se obtienen los parámetros del índice. El nombre de índice distingue entre mayúsculas y minúsculas, y debe aparecer listado en la vista de catálogo SYSCAT.INDEXES para el esquema *esquema_índice*.

esquema_tabla
Valor de tipo VARCHAR(128) que identifica el esquema donde está contenida la tabla cuyo nombre no calificado es *nombre_tabla*. El nombre de esquema distingue entre mayúsculas y minúsculas, y debe aparecer listado en la vista de catálogo SYSCAT.SCHEMATA.

Si este parámetro tiene un valor nulo, se utiliza el valor del registro especial CURRENT SCHEMA como nombre de esquema para el índice espacial.

nombre_tabla
Valor de tipo VARCHAR(128) que contiene el nombre no calificado de la tabla que tiene la columna espacial *nombre_columna*. El nombre de tabla distingue entre mayúsculas y minúsculas, y debe aparecer listado en la vista de catálogo SYSCAT.TABLES para el esquema *esquema_tabla*.

nombre_columna
Valor de tipo VARCHAR(128) que identifica la columna de la tabla *esquema_tabla.nombre_tabla* para la cual se obtienen los parámetros del índice espacial definido en esa columna. El nombre de columna distingue entre mayúsculas y minúsculas, y debe aparecer listado en la vista de catálogo SYSCAT.COLUMNS para la tabla *esquema_tabla.nombre_tabla*.

Si no existe ningún índice espacial definido en la columna, se emite un error (SQLSTATE 38SQ0).

número_tamaño_retícula
Valor de tipo DOUBLE que identifica el parámetro para el cual se obtiene su valor o valores.

Si este valor es menor que 1 o mayor que 3, se emite un error (SQLSTATE 38SQ1).

Tipo de retorno:

DOUBLE (si está especificado *número_tamaño_retícula*)

Si *número_tamaño_retícula* no está especificado, se devuelve una tabla que contiene dos columnas: ORDINAL y VALUE. La columna ORDINAL es de tipo INTEGER, y la columna VALUE es de tipo DOUBLE.

Si los parámetros se obtienen para un índice reticular, la columna ORDINAL contiene los valores 1, 2 y 3 para el primer, segundo y tercer tamaño de retícula, respectivamente. La columna VALUE contiene los tamaños de retícula.

La columna VALUE contiene los valores respectivos para cada parámetro.

Ejemplos:

Este código crea una tabla con una columna espacial y un índice espacial.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sch.offices (name VARCHAR(30), location ST_Point )

 CREATE INDEX sch.idx ON sch.offices(location)
      EXTEND USING db2gse.spatial_index(1e0, 10e0, 1000e0)
```

Se puede utilizar la función ST_GetIndexParms para obtener los valores de los parámetros que se utilizaron al crear el índice espacial.

Ejemplo 1:

Este ejemplo muestra cómo obtener por separado los tres tamaños de retícula para un índice reticular espacial, mediante la especificación explícita del parámetro a devolver, identificado por su número.

```
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 1)
```

Resultados:

```
1
------------------------
  +1.00000000000000E+000

VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 2)
```

Resultados:

```
1
------------------------
  +1.00000000000000E+001
VALUES ST_GetIndexParms('SCH', 'IDX', 3)
```

Resultados:

```
1
------------------------
  +1.00000000000000E+003
```

Ejemplo 2:

Este ejemplo muestra cómo obtener todos los parámetros de un índice reticular espacial. La función ST_GetIndexParms obtiene una tabla que indica el número de parámetro y el correspondiente tamaño de retícula.

```
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION') ) AS t
```

Resultados:

```
ORDINAL     VALUE
----------- ----------------------
          1   +1.00000000000000E+000
          2   +1.00000000000000E+001
          3   +1.00000000000000E+003
```

```
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'IDX') ) AS t
```

Resultados:

```
ORDINAL     VALUE
----------- ----------------------
          1   +1.00000000000000E+000
          2   +1.00000000000000E+001
          3   +1.00000000000000E+003
```

Conceptos relacionados:

- "Índices reticulares espaciales" en la página 104

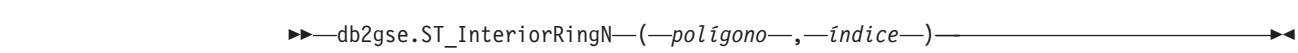
ST_InteriorRingN

ST_InteriorRingN toma un polígono y un índice como parámetros de entrada y devuelve el anillo interior identificado por dicho índice en forma de cadena lineal. Los anillos interiores se organizan según las normas definidas por las rutinas internas de verificación de la geometría.

Si el polígono proporcionado es un valor nulo o está vacío, o si no tiene anillos interiores, se devuelve un valor nulo. Si el índice es menor que 1 o mayor que el número de anillos interiores del polígono, se devuelve un valor nulo y se emite una condición de aviso (1HS1).

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

polígono
: Valor de tipo ST_Polygon que representa la geometría a partir de la cual se devuelve el anillo interior identificado por *índice*.

índice
: Valor de tipo INTEGER que identifica el anillo interior *n* que se devuelve. Si el anillo interior identificado por *índice* no existe, se emite una condición de aviso (01HS1).

Tipo de retorno:

db2gse.ST_Curve

Ejemplo:

El ejemplo siguiente crea un polígono con dos anillos interiores. A continuación, se invoca a ST_InteriorRingN para obtener el segundo anillo interior.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys VALUES
       (1, ST_Polygon('polygon((40 120, 90 120, 90 150, 40 150, 40 120),
                      (50 130, 60 130, 60 140, 50 140, 50 130),
                      (70 130, 80 130, 80 140, 70 140, 70 130))' ,0))


SELECT id, CAST(ST_AsText(ST_InteriorRingN(geometry, 2)) as VARCHAR(180))
       Interior_Ring
FROM sample_polys
```

Resultados:

```
ID          INTERIOR_RING
----------- ----------------------------------------------------------------
      1 LINESTRING ( 70.00000000 130.00000000, 70.00000000 140.00000000,
80.00000000 140.00000000, 80.00000000 130.00000000, 70.00000000 130.00000000)
```

Información relacionada:

- "ST_ExteriorRing" en la página 397
- "ST_NumInteriorRing" en la página 468

ST_Intersection

ST_Intersection toma dos geometrías como parámetros de entrada y devuelve la geometría que es la intersección de estas dos geometrías. La intersección es la parte común de la primera geometría y la segunda geometría. La geometría resultante se representa según el sistema de referencia espacial de la primera geometría.

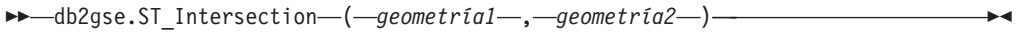
Si es posible, el tipo específico de la geometría devuelta será ST_Point, ST_LineString o ST_Polygon. Por ejemplo, la intersección de un punto y un polígono está vacía o es un único punto, que se representa como ST_MultiPoint.

Si cualquiera de las dos geometrías es un valor nulo, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría1
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la primera geometría para calcular la intersección con *geometría2*.

geometría2
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la segunda geometría para calcular la intersección con *geometría1*.

Para los datos geodésicos, ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

Tipo de retorno:

db2gse.ST_Geometry

La dimensión de la geometría devuelta es la de la entrada con la dimensión inferior, con la excepción de las cadenas lineales de los datos geodésicos. Para los datos geodésicos, la dimensión de la intersección de dos cadenas lineales es 0 (en otras palabras, la intersección es un punto o un multipunto).

Ejemplo:

En los ejemplos siguientes, los resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo crea varias geometrías diferentes y luego determina la intersección (si existe) con la primera geometría.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (1, ST_Geometry('polygon((30 30, 30 50, 50 50, 50 30, 30 30))' ,0))

INSERT INTO sample_geoms VALUES
       (2, ST_Geometry('polygon((20 30, 30 30, 30 40, 20 40, 20 30))' ,0))

INSERT INTO sample_geoms VALUES
       (3, ST_Geometry('polygon((40 40, 40 60, 60 60, 60 40, 40 40))' ,0))

INSERT INTO sample_geoms VALUES
       (4, ST_Geometry('linestring(60 60, 70 70)' ,0))

INSERT INTO sample_geoms VALUES
       (5, ST_Geometry('linestring(30 30, 60 60)' ,0))

SELECT a.id, b.id, CAST(ST_AsText(ST_Intersection(a.geometry, b.geometry))
   as VARCHAR(150)) Intersection
FROM sample_geoms a, sample_geoms b
WHERE a.id = 1
```

Resultados:

```
ID          ID          INTERSECTION
----------- ----------- ------------------------------------------------
          1           1 POLYGON (( 30.00000000 30.00000000, 50.00000000
30.00000000, 50.00000000 50.00000000, 30.00000000 50.00000000, 30.00000000
30.00000000))

          1           2 LINESTRING ( 30.00000000 40.00000000, 30.00000000
30.00000000)

          1           3 POLYGON (( 40.00000000 40.00000000, 50.00000000
40.00000000, 50.00000000 50.00000000, 40.00000000 50.00000000, 40.00000000
40.00000000))

          1           4 POINT EMPTY

          1           5 LINESTRING ( 30.00000000 30.00000000, 50.00000000
50.00000000)

    5 registro(s) seleccionados.
```

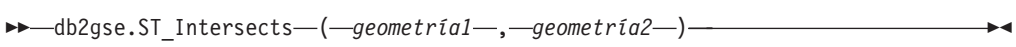
ST_Intersects

ST_Intersects toma dos geometrías como parámetros de entrada y devuelve un 1 si dichas geometrías forman intersección. Si las geometrías no forman intersección, se devuelve un 0 (cero).

Si cualquiera de las geometrías es un valor nulo o está vacía, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

Sintaxis:



Parámetro:

geometría1
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para probar la intersección con *geometría2*.

geometría2
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para probar la intersección con *geometría1*.

Restricciones: Para los datos geodésicos, ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

Tipo de retorno:

INTEGER

Ejemplo:

Las siguientes sentencias crean y llenan las tablas SAMPLE_GEOMETRIES1 y SAMPLE_GEOMETRIES2.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries1(id SMALLINT, spatial_type varchar(13),
    geometry ST_GEOMETRY);
CREATE TABLE sample_geometries2(id SMALLINT, spatial_type varchar(13),
    geometry ST_GEOMETRY);

INSERT INTO sample_geometries1(id, spatial_type, geometry)
VALUES
    ( 1, 'ST_Point', ST_Point('point(550 150)', 1) ),
    (10, 'ST_LineString', ST_LineString('linestring(800 800, 900 800)', 1)),
    (20, 'ST_Polygon', ST_Polygon('polygon((500 100, 500 200, 700 200,
          700 100, 500 100))', 1) )

INSERT INTO sample_geometries2(id, spatial_type, geometry)
VALUES
    (101, 'ST_Point', ST_Point('point(550 150)', 1) ),
    (102, 'ST_Point', ST_Point('point(650 200)', 1) ),
    (103, 'ST_Point', ST_Point('point(800 800)', 1) ),
    (110, 'ST_LineString', ST_LineString('linestring(850 250, 850 850)', 1)),
```

```
    (120, 'ST_Polygon', ST_Polygon('polygon((650 50, 650 150, 800 150,
           800 50, 650 50))', 1)),
    (121, 'ST_Polygon', ST_Polygon('polygon((20 20, 20 40, 40 40, 40 20,
           20 20))', 1) )
```

La sentencia SELECT siguiente determina si las diversas geometrías contenidas en las tablas SAMPLE_GEOMTRIES1 y SAMPLE_GEOMTRIES2 forman intersección.

```
SELECT   sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
         sg2.id AS sg2_id, sg2.spatial_type AS sg2_type,
         CASE ST_Intersects(sg1.geometry, sg2.geometry)
            WHEN 0 THEN 'Las geometrías no forman intersección'
            WHEN 1 THEN 'Las geometrías forman intersección'
         END AS intersects
FROM     sample_geometries1 sg1, sample_geometries2 sg2
ORDER BY sg1.id
```

Resultados:

```
SG1_ID SG1_TYPE      SG2_ID SG2_TYPE      INTERSECTS
------ ------------- ------ ------------- --------------------------
     1 ST_Point         101 ST_Point      Las geometrías forman intersección
     1 ST_Point         102 ST_Point      Las geometrías no forman intersección
     1 ST_Point         103 ST_Point      Las geometrías no forman intersección
     1 ST_Point         110 ST_LineString Las geometrías no forman intersección
     1 ST_Point         120 ST_Polygon    Las geometrías no forman intersección
     1 ST_Point         121 ST_Polygon    Las geometrías no forman intersección
    10 ST_LineString    101 ST_Point      Las geometrías no forman intersección
    10 ST_LineString    102 ST_Point      Las geometrías no forman intersección
    10 ST_LineString    103 ST_Point      Las geometrías forman intersección
    10 ST_LineString    110 ST_LineString Las geometrías forman intersección
    10 ST_LineString    120 ST_Polygon    Las geometrías no forman intersección
    10 ST_LineString    121 ST_Polygon    Las geometrías no forman intersección
    20 ST_Polygon       101 ST_Point      Las geometrías forman intersección
    20 ST_Polygon       102 ST_Point      Las geometrías forman intersección
    20 ST_Polygon       103 ST_Point      Las geometrías no forman intersección
    20 ST_Polygon       110 ST_LineString Las geometrías no forman intersección
    20 ST_Polygon       120 ST_Polygon    Las geometrías forman intersección
    20 ST_Polygon       121 ST_Polygon    Las geometrías no forman intersección
```

Información relacionada:

• "Funciones que comparan elementos geográficos" en la página 316

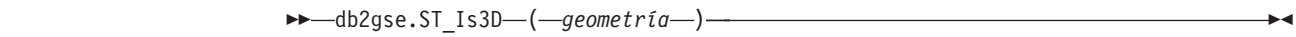
ST_Is3d

ST_Is3d toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría tiene coordenadas Z. En caso contrario, se devuelve un 0 (cero).

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría en la que se va a verificar si existen coordenadas Z.

Tipo de retorno:

INTEGER

Ejemplo:

Este ejemplo crea varias geometrías con y sin coordenadas Z y M (medidas). Luego, se utiliza ST_Is3d para determinar cuál de las geometrías contiene coordenadas Z.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (1, ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
       (2, ST_Geometry('polygon((40 120, 90 120, 90 150, 40 150, 40 120))' ,0))

INSERT INTO sample_geoms VALUES
       (3, ST_Geometry('multipoint m (10 10 5, 50 10 6, 10 30 8)' ,0))

INSERT INTO sample_geoms VALUES
       (4, ST_Geometry('linestring z (10 10 166, 20 10 168)',0))

INSERT INTO sample_geoms VALUES
       (5, ST_Geometry('point zm (10 10 16 30)' ,0))


SELECT id, ST_Is3d(geometry)  Is_3D
FROM sample_geoms
```

Resultados:

```
ID          IS_3D
----------- -----------
          1           0
          2           0
          3           0
          4           1
          5           1
```

ST_IsClosed

ST_IsClosed toma una curva o multicurva como parámetro de entrada y devuelve un 1 si dicha curva o multicurva es cerrada. En caso contrario, se devuelve un 0 (cero).

Una curva es cerrada si el punto inicial y el punto final son iguales. Si la curva tiene coordenadas Z, las coordenadas Z del punto inicial y final deben ser iguales. En caso contrario, los puntos no se consideran iguales y la curva no estará cerrada. Una multicurva es cerrada si cada una de sus curvas es cerrada.

Si la curva o multicurva proporcionada está vacía, se devuelve un 0 (cero). Si es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

curva Valor de tipo ST_Curve o ST_MultiCurve o uno de sus subtipos que representa la curva o multicurva que se va a verificar.

Tipo de retorno:

INTEGER

Ejemplos:

Ejemplo 1:

Este ejemplo crea varias cadenas lineales. Las dos últimas cadenas lineales tienen las mismas coordenadas X e Y, pero una cadena lineal contiene coordenadas Z variables que hacen que la cadena lineal no sea cerrada, y las demás cadenas lineales contienen coordenadas M variables (medidas) que no afectan en el hecho de si la cadena lineal es cerrada.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, geometry ST_Linestring)

INSERT INTO sample_lines VALUES
       (1, ST_Linestring('linestring EMPTY',0))

INSERT INTO sample_lines VALUES
       (2, ST_Linestring('linestring(10 10, 20 10, 20 20)' ,0))

INSERT INTO sample_lines VALUES
       (3, ST_Linestring('linestring(10 10, 20 10, 20 20, 10 10)' ,0))

INSERT INTO sample_lines VALUES
       (4, ST_Linestring('linestring m(10 10 1, 20 10 2, 20 20 3,
        10 10 4)' ,0))

INSERT INTO sample_lines VALUES
       (5, ST_Linestring('linestring z(10 10 5, 20 10 6, 20 20 7,
        10 10 8)' ,0))

SELECT id, ST_IsClosed(geometry)  Is_Closed
FROM sample_lines
```

Resultados:

```
ID          IS_CLOSED
----------- -----------
          1           0
          2           0
          3           1
          4           1
          5           0
```

Ejemplo 2:

Este ejemplo crea dos multilíneas. ST_IsClosed se utiliza para determinar si las multilíneas están cerradas. La primera multilínea no es cerrada, aunque todas las curvas forman conjuntamente un bucle cerrado completo. Esto es debido a que cada curva individual no es cerrada.

La segunda multilínea es cerrada, pues cada curva individual es cerrada.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mlines (id INTEGER, geometry ST_MultiLinestring)
INSERT INTO sample_mlines VALUES
       (6, ST_MultiLinestring('multilinestring((10 10, 20 10, 20 20),
                                               (20 20, 30 20, 30 30),
                                               (30 30, 10 30, 10 10))',0))

INSERT INTO sample_mlines VALUES
       (7, ST_MultiLinestring('multilinestring((10 10, 20 10, 20 20, 10 10 ),
                                                (30 30, 50 30, 50 50,
                                                 30 30 ))',0))

SELECT id, ST_IsClosed(geometry)  Is_Closed
FROM sample_mlines
```

Resultados:

```
ID          IS_CLOSED
----------- -----------
          6           0
          7           1
```

ST_IsEmpty

ST_IsEmpty toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría está vacía. En caso contrario, se devuelve un 0 (cero). Una geometría está vacía si no tiene ningún punto que la defina.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se va a verificar.

Tipo de retorno:

INTEGER

Ejemplo:

El código siguiente crea tres geometrías y luego determina si están vacías.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (1, ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
       (2, ST_Geometry('polygon((40 120, 90 120, 90 150, 40 150, 40 120))' ,0))
```

```
INSERT INTO sample_geoms VALUES
       (3, ST_Geometry('multipoint m (10 10 5, 50 10 6, 10 30 8)' ,0))

INSERT INTO sample_geoms VALUES
       (4, ST_Geometry('linestring z (10 10 166, 20 10 168)',0))

INSERT INTO sample_geoms VALUES
       (5, ST_Geometry('point zm (10 10 16 30)' ,0))


SELECT id, ST_IsEmpty(geometry)  Is_Empty
FROM sample_geoms
```

Resultados:

```
ID          IS_EMPTY
----------- -----------
          1           1
          2           0
          3           0
          4           0
          5           0
```

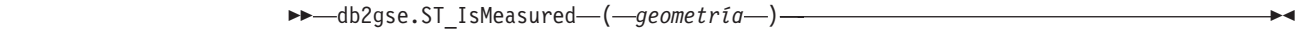
ST_IsMeasured

ST_IsMeasured toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría tiene coordenadas M (medidas). En caso contrario, se devuelve un 0 (cero).

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría en la que se va a verificar si existen coordenadas M (medidas).

Tipo de retorno:

INTEGER

Ejemplo:

Este ejemplo crea varias geometrías con y sin coordenadas Z y M (medidas). Luego, se utiliza ST_IsMeasured para determinar cuál de las geometrías contiene medidas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
```

```
       (1, ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
       (2, ST_Geometry('polygon((40 120, 90 120, 90 150, 40 150, 40 120))' ,0))

INSERT INTO sample_geoms VALUES
       (3, ST_Geometry('multipoint m (10 10 5, 50 10 6, 10 30 8)' ,0))

INSERT INTO sample_geoms VALUES
       (4, ST_Geometry('linestring z (10 10 166, 20 10 168)',0))

INSERT INTO sample_geoms VALUES
       (5, ST_Geometry('point zm (10 10 16 30)' ,0))


SELECT id, ST_IsMeasured(geometry)  Is_Measured
FROM sample_geoms
```

Resultados:

```
ID          IS_MEASURED
----------- -----------
          1           0
          2           0
          3           1
          4           0
          5           1
```

ST_IsRing

ST_IsRing toma una curva como parámetro de entrada y devuelve un 1 si es un anillo. En caso contrario, se devuelve un 0 (cero). Una curva es un anillo si es simple y cerrada.

Si la curva proporcionada está vacía, se devuelve un 0 (cero). Si es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

curva Valor de tipo ST_Curve o uno de sus subtipos que representa la curva que se va a verificar.

Tipo de retorno:

INTEGER

Ejemplos:

Este ejemplo crea cuatro cadenas lineales. Luego, se utiliza ST_IsRing para comprobar si las cadenas lineales son anillos. La última cadena lineal no es considerada un anillo aunque es cerrada, debido a que forma intersección consigo misma.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, geometry ST_Linestring)

INSERT INTO sample_lines VALUES
       (1, ST_Linestring('linestring EMPTY',0))

INSERT INTO sample_lines VALUES
       (2, ST_Linestring('linestring(10 10, 20 10, 20 20)' ,0))

INSERT INTO sample_lines VALUES
       (3, ST_Linestring('linestring(10 10, 20 10, 20 20, 10 10)' ,0))

INSERT INTO sample_lines VALUES
       (4, ST_Linestring('linestring(10 10, 20 10, 10 20, 20 20, 10 10)' ,0))


SELECT id, ST_IsClosed(geometry) Is_Closed, ST_IsRing(geometry)  Is_Ring
FROM sample_lines
```

Resultados:

```
ID          IS_CLOSED   IS_RING
----------- ----------- -----------
          1           1           0
          2           0           0
          3           1           1
          4           1           0
```

Información relacionada:

- "ST_IsClosed" en la página 420
- "ST_IsSimple" en la página 425

ST_IsSimple

ST_IsSimple toma una geometría como parámetro de entrada y devuelve un 1 si dicha geometría es simple. En caso contrario, se devuelve un 0 (cero).

Los puntos, las superficies y multisuperficies son siempre simples. Una curva es simple si no atraviesa el mismo punto dos veces; un multipunto es simple si no contiene dos puntos iguales; y una multicurva es simple si todas sus curvas son simples y las únicas intersecciones se producen en puntos que se encuentran en el límite de las curvas que forman la multicurva.

Si la geometría proporcionada está vacía, se devuelve un 1. Si la geometría es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se va a verificar.

Tipo de retorno:

INTEGER

Ejemplos:

Este ejemplo crea varias geometrías y comprueba si son simples. La geometría cuyo ID es 4 no se considera que sea simple porque contiene más de un punto que es el mismo. La geometría cuyo ID es 6 no se considera que sea simple porque la cadena lineal forma intersección consigo misma.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (1, ST_Geometry('point EMPTY' ,0))

INSERT INTO sample_geoms VALUES
       (2, ST_Geometry('point (21 33)' ,0))

INSERT INTO sample_geoms VALUES
       (3, ST_Geometry('multipoint(10 10, 20 20, 30 30)' ,0))

INSERT INTO sample_geoms VALUES
       (4, ST_Geometry('multipoint(10 10, 20 20, 30 30, 20 20)' ,0))

INSERT INTO sample_geoms VALUES
       (5, ST_Geometry('linestring(60 60, 70 60, 70 70)' ,0))

INSERT INTO sample_geoms VALUES
       (6, ST_Geometry('linestring(20 20, 30 30, 30 20, 20 30  )' ,0))

INSERT INTO sample_geoms VALUES
       (7, ST_Geometry('polygon((40 40, 50 40, 50 50, 40 40  ))' ,0))

SELECT id, ST_IsSimple(geometry) Is_Simple
FROM sample_geoms
```

Resultados:

```
ID          IS_SIMPLE
----------- -----------
          1           1
          2           1
          3           1
          4           0
          5           1
          6           0
          7           1
```

ST_IsValid

ST_IsValid toma una geometría como parámetro de entrada y devuelve un 1 si la geometría es válida. En caso contrario, se devuelve un 0 (cero). Una geometría es válida sólo si todos los atributos contenidos en el tipo estructurado son coherentes con la representación interna de los datos de la geometría, y si la representación interna no está dañada.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos.

Tipo de retorno:

INTEGER

Ejemplo:

Este ejemplo crea varias geometrías y utiliza ST_IsValid para comprobar su validez. Todas las geometrías son válidas porque la rutinas constructoras, tales como ST_Geometry, no permiten la creación de geometrías no válidas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
       (1, ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
       (2, ST_Geometry('polygon((40 120, 90 120, 90 150, 40 150, 40 120))' ,0))

INSERT INTO sample_geoms VALUES
       (3, ST_Geometry('multipoint m (10 10 5, 50 10 6, 10 30 8)' ,0))

INSERT INTO sample_geoms VALUES
       (4, ST_Geometry('linestring z (10 10 166, 20 10 168)',0))

INSERT INTO sample_geoms VALUES
       (5, ST_Geometry('point zm (10 10 16 30)' ,0))


SELECT id, ST_IsValid(geometry)  Is_Valid
FROM sample_geoms
```

Resultados:

```
ID          IS_VALID
----------- -----------
          1           1
          2           1
          3           1
          4           1
          5           1
```

ST_Length

ST_Length toma como parámetros de entrada una curva o multicurva y, opcionalmente, una unidad de medida, y devuelve la longitud de la curva o multicurva proporcionada, expresada en la unidad de medida proporcionada o en la unidad por omisión.

Si la curva o multicurva proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

curva Valor de tipo ST_Curve o ST_MultiCurve que representa las curvas para las que se obtiene la longitud.

unidad Valor de tipo VARCHAR(128) que identifica las unidades utilizadas para medir la longitud. Las unidades de medida soportadas están listadas en la vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE.

Si se omite el parámetro *unidad*, se siguen las normas siguientes para determinar la unidad utilizada para medir la longitud:

- Si la *curva* está en un sistema de coordenadas proyectadas o geocéntricas, se utiliza la unidad lineal correspondiente a este sistema de coordenadas por omisión.
- Si la *curva* está en un sistema de coordenadas geográficas pero no en un sistema de referencia espacial (SRS) geodésico, se utiliza la unidad angular correspondiente a este sistema de coordenadas por omisión.
- Si la *curva* está en un SRS geodésico, la unidad de medida por omisión será el metro.

Restricciones para las conversiones de unidades: Se devuelve un error (SQLSTATE 38SU4) si se cumple alguna de las siguientes condiciones:

- La *curva* está en un sistema de coordenadas no especificado y se especifica el parámetro *unidad*.
- La *curva* está en un sistema de coordenadas proyectadas y se especifica una unidad angular.
- La *curva* está en un sistema de coordenadas geográficas pero no en un SRS geodésico y se ha especificado una unidad lineal.
- La *curva* está en un SRS geodésico y se especifica una unidad angular.

Tipo de retorno:

DOUBLE

Ejemplos:

Las siguientes sentencias de SQL crean la tabla SAMPLE_GEOMETRIES e insertan una línea y una multilínea en la tabla.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id SMALLINT, spatial_type varchar(20),
    geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, spatial_type, geometry)
VALUES
    (1110, 'ST_LineString', ST_LineString('linestring(50 10, 50 20)', 1)),
```

```
    (1111, 'ST_MultiLineString', ST_MultiLineString('multilinestring
                            ((33 2, 34 3, 35 6),
                            (28 4, 29 5, 31 8, 43 12),
                            (39 3, 37 4, 36 7))', 1))
```

Ejemplo 1:

La sentencia SELECT siguiente calcula la longitud de la línea contenida en la tabla SAMPLE_GEOMTRIES.

```
SELECT id, spatial_type, cast(ST_Length(geometry..ST_ToLineString)
   AS DECIMAL(7, 2)) AS "Line Length"
FROM   sample_geometries
WHERE  id = 1110
```

Resultados:

```
ID     SPATIAL_TYPE         Line Length
------ -------------------- -----------
  1110 ST_LineString              10.00
```

Ejemplo 2:

La sentencia SELECT siguiente calcula la longitud de la multilínea contenida en la tabla SAMPLE_GEOMTRIES.

```
SELECT id, spatial_type, ST_Length(ST_ToMultiLine(geometry))
   AS multiline_length
FROM   sample_geometries
WHERE  id = 1111
```

Resultados:

```
ID     SPATIAL_TYPE         MULTILINE_LENGTH
 ------ -------------------- ------------------------
   1111 ST_MultiLineString     +2.76437123387202E+001
```

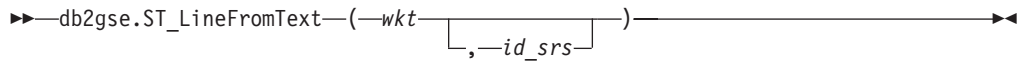
ST_LineFromText

ST_LineFromText toma como parámetros de entrada una representación de texto convencional de una cadena lineal y, opcionalmente, un identificador de sistemas de referencia espacial, y devuelve la cadena lineal correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La versión preferida para esta función es ST_LineString.

Sintaxis:



Parámetro:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la cadena lineal resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la cadena lineal resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite un error (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_LineString

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente muestra cómo utilizar la función ST_LineFromText para crear e insertar una línea a partir de una representación de texto convencional (WKT) de una línea. Las filas se insertan en la tabla SAMPLE_LINES con un ID y un valor de línea en el sistema de referencia espacial 1 utilizando una representación WKT.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_lines(id SMALLINT, geometry ST_LineString)

INSERT INTO sample_lines(id, geometry)
VALUES
    (1110, ST_LineFromText('linestring(850 250, 850 850)', 1) ),
    (1111, ST_LineFromText('linestring empty', 1) )

SELECT id, cast(geometry..ST_AsText AS varchar(75)) AS linestring
FROM   sample_lines
```

Resultados:

```
ID     LINESTRING
------ ---------------------------------------------------------------------
  1110 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)
  1111 LINESTRING EMPTY
```

Información relacionada:

- "Representación de texto convencional (WKT)" en la página 529

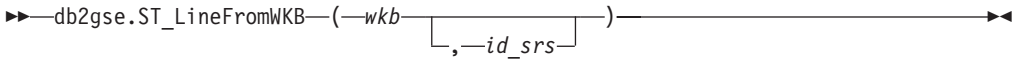
ST_LineFromWKB

ST_LineFromWKB toma como parámetros de entrada una representación binaria convencional de una cadena lineal y, opcionalmente, un identificador de sistemas de referencia espacial, y devuelve la cadena lineal correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La versión preferida para esta función es ST_LineString.

Sintaxis:



Parámetro:

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la cadena lineal resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la cadena lineal resultante.

Si se omite el parámetro ***id_srs***, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite un error (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_LineString

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

El código siguiente utiliza la función ST_LineFromWKB para crear e insertar una línea a partir de una representación binaria convencional. La fila se inserta en la tabla SAMPLE_LINES con un ID y una línea en el sistema de referencia espacial 1 utilizando una representación WKB.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_lines(id SMALLINT, geometry ST_LineString, wkb BLOB(32k))

INSERT INTO sample_lines(id, geometry)
VALUES
    (1901, ST_LineString('linestring(850 250, 850 850)', 1) ),
    (1902, ST_LineString('linestring(33 2, 34 3, 35 6)', 1) )

UPDATE sample_lines AS temp_correlated
SET    wkb = geometry..ST_AsBinary
WHERE  id = temp_correlated.id

SELECT id, cast(ST_LineFromWKB(wkb)..ST_AsText AS varchar(90)) AS line
FROM   sample_lines
```

Resultados:

```
ID     LINE
------ ------------------------------------------------------------------
  1901 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)

  1902 LINESTRING ( 33.00000000 2.00000000, 34.00000000 3.00000000,
35.00000000 6.00000000)
```

Información relacionada:

- "Representación binaria convencional (WKB)" en la página 534

ST_LineString

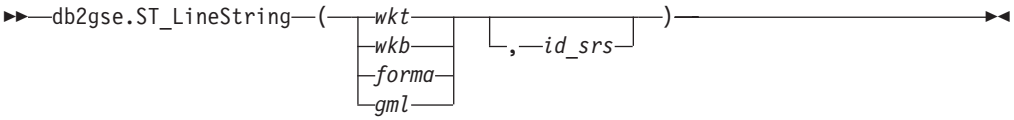
ST_LineString crea una cadena lineal partir de uno de los datos de entrada siguientes:

- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma ESRI
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenida la cadena lineal resultante.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma ESRI o la representación GML es nula, se devuelve un valor nulo.

Sintaxis:



Parámetro:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional del polígono resultante.

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional del polígono resultante.

forma Valor de tipo BLOB(2G) que contiene la representación de forma ESRI del polígono resultante.

gml Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), del polígono resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al polígono resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite un error (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_LineString

Ejemplos:

El código siguiente utiliza la función ST_LineString para crear e insertar una línea a partir de una representación de texto convencional (WKT) de una línea o de una representación binaria convencional.

El ejemplo siguiente inserta una fila en la tabla SAMPLE_LINES, con un ID y una línea en el sistema de referencia espacial 1, en una representación WKT y GML.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_lines(id SMALLINT, geometry ST_LineString)

INSERT INTO sample_lines(id, geometry)
VALUES
    (1110, ST_LineString('linestring(850 250, 850 850)', 1) ),
    (1111, ST_LineString('<gml:LineString srsName=";EPSG:4269";><gml:coord>
                          <gml:X>90</gml:X><gml:Y>90</gml:Y>
                          </gml:coord><gml:coord><gml:X>100</gml:X>
                          <gml:Y>100</gml:Y></gml:coord>
                          </gml:LineString>', 1) )

SELECT id, cast(geometry..ST_AsText AS varchar(75)) AS linestring
FROM   sample_lines
```

Resultados:

```
ID     LINESTRING
------ ---------------------------------------------------------------
  1110 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)
  1111 LINESTRING ( 90.00000000 90.00000000, 100.00000000 100.00000000)
```

Información relacionada:

- "Funciones espaciales que convierten valores de geometría en formatos de intercambio de datos" en la página 307
- "Representación de texto convencional (WKT)" en la página 529

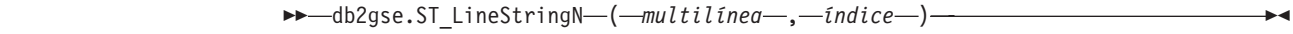
ST_LineStringN

ST_LineStringN toma un multilínea y un índice como parámetros de entrada y devuelve la cadena lineal identificada por el índice. La cadena lineal resultante se representa según el sistema de referencia espacial de la multilínea proporcionada.

Si la multilínea proporcionada tiene un valor nulo o está vacía, o si el índice es menor que 1 o mayor que el número de cadenas lineales, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

multilínea
: Valor de tipo ST_MultiLineString que representa la multilínea a partir de la cual se obtiene la cadena lineal identificada por *índice*.

índice
: Valor de tipo INTEGER que identifica la cadena lineal *n* que se obtendrá a partir de *multilínea*.

  Si *índice* es menor que 1 o mayor que el número de cadenas lineales de *multilínea*, se devuelve un valor nulo y se emite una condición de aviso (SQLSTATE 01HS0).

Tipo de retorno:

db2gse.ST_LineString

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

La sentencia SELECT selecciona la segunda geometría contenida en una multilínea de la tabla SAMPLE_MLINES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_mlines (id INTEGER,
  geometry ST_MULTILINESTRING)

INSERT INTO sample_mlines(id, geometry)
VALUES
    (1110, ST_MultiLineString('multilinestring
                              ((33 2, 34 3, 35 6),
                              (28 4, 29 5, 31 8, 43 12),
                              (39 3, 37 4, 36 7))', 1) ),
    (1111, ST_MLineFromText('multilinestring(
                            (61 2, 64 3, 65 6),
                            (58 4, 59 5, 61 8),
                            (69 3, 67 4, 66 7, 68 9))', 1) )

SELECT id, cast(ST_LineStringN(geometry, 2)..ST_AsText
  AS varchar(110)) AS second_linestring
FROM   sample_mlines
```

Resultados:

```
ID          SECOND_LINESTRING
----------- ----------------------------------------------------------------------
1110         LINESTRING ( 28.00000000 4.00000000, 29.00000000
   5.00000000, 31.00000000 8.00000000, 43.00000000 12.00000000)

1111         LINESTRING ( 58.00000000 4.00000000, 59.00000000
   5.00000000, 61.00000000 8.00000000)
```

Información relacionada:

- "ST_NumLineStrings" en la página 469

ST_M

ST_M puede actuar de una de estas maneras:

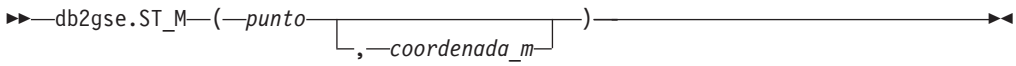
- Tomar un punto como parámetro de entrada y devolver su coordenada M (medida)
- Tomar como parámetros de entrada un punto y una coordenada M y devolver el propio punto junto con su coordenada M expresada en la medida proporcionada, aunque el punto especificado no tenga ninguna coordenada M.

Si la coordenada M especificada es nula, la coordenada M del punto se elimina.

Si el punto especificado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

punto Valor de tipo ST_Point para el cual se obtiene o modifica la coordenada M.

coordenada_m
Valor de tipo DOUBLE que representa la nueva coordenada M del *punto*.

Si *coordenada_m* es nulo, la coordenada M se elimina de *punto*.

Tipos de retorno:

- DOUBLE, si no se especifica *coordenada_m*
- db2gse.ST_Point, si se especifica *coordenada_m*

Ejemplos:

Estos ejemplos muestran el uso de la función ST_M. Se crean tres puntos y luego se insertan en la tabla SAMPLE_POINTS. Todos ellos están en el sistema de referencia espacial cuyo ID es 1.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points
  VALUES (1, ST_Point (2, 3, 32, 5, 1))

INSERT INTO sample_points
  VALUES (2, ST_Point (4, 5, 20, 4, 1))

INSERT INTO sample_points
  VALUES (3, ST_Point (3, 8, 23, 7, 1))
```

Ejemplo 1:

Este ejemplo determina la coordenada M de los puntos contenidos en la tabla SAMPLE_POINTS.

```
SELECT id, ST_M (geometry) M_COORD
  FROM sample_points
```

Resultados:

```
ID          M_COORD
----------- ------------------------
          1 +5.00000000000000E+000
          2 +4.00000000000000E+000
          3 +7.00000000000000E+000
```

Ejemplo 2:

Este ejemplo obtiene uno de los puntos junto con su coordenada M establecida en el valor 40.

```
SELECT id, CAST (ST_AsText (ST_M (geometry, 40) )
  AS VARCHAR(60) ) M_COORD_40
  FROM sample_points
  WHERE id=3
```

Resultados:

```
ID         M_COORD_40
---------- ----------------------------------------------------
         3 POINT ZM (3.00000000 8.00000000 23.00000000 40.00000000)
```

Información relacionada:

- "ST_X" en la página 515
- "ST_Y" en la página 517
- "ST_Z" en la página 518

ST_MaxM

ST_MaxM toma una geometría como parámetro de entrada y devuelve su coordenada M máxima.

Si la geometría proporcionada es un valor nulo o está vacía, o si no tiene coordenadas M, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos para el que se devuelve la coordenada M máxima.

Tipo de retorno:

DOUBLE

Ejemplos:

Estos ejemplos muestran el uso de la función ST_MaxM. Se crean tres polígonos y luego se insertan en la tabla SAMPLE_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                      110 140 22 3,
                                      120 130 26 4,
                                      110 120 20 3))', 0) )

INSERT INTO sample_polys
  VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                      0 4 35 9,
                                      5 4 32 12,
                                      5 0 31 5,
                                      0 0 40 7))', 0) )

INSERT INTO sample_polys
  VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                      8 4 10 12,
                                      9 4 12 11,
                                      12 13 10 16))', 0) )
```

Ejemplo 1:

Este ejemplo determina la coordenada M máxima de cada polígono contenido en SAMPLE_POLYS.

```
SELECT id, CAST ( ST_MaxM(geometry) AS INTEGER) MAX_M
  FROM sample_polys
```

Resultados:

```
ID          MAX_M
----------- ------------
          1            4
          2           12
          3           16
```

Ejemplo 2:

Este ejemplo determina la coordenada M máxima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxM(geometry) ) AS INTEGER) OVERALL_MAX_M
  FROM sample_polys
```

Resultados:

```
OVERALL_MAX_M
--------------
           16
```

Conceptos relacionados:

- "ST_MaxX" en la página 437

Información relacionada:

- "ST_MaxY" en la página 439
- "ST_MaxZ" en la página 440
- "ST_MinM" en la página 447

ST_MaxX

ST_MaxX toma una geometría como parámetro de entrada y devuelve su coordenada X máxima.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos para el que se devuelve la coordenada X máxima.

Tipo de retorno:

DOUBLE

Ejemplos:

Estos ejemplos muestran el uso de la función ST_MaxX. Se crean tres polígonos y luego se insertan en la tabla SAMPLE_POLYS. El tercer ejemplo muestra cómo utilizar todas las funciones que obtienen las coordenadas máxima y mínima para evaluar el rango espacial de las geometrías contenidas en una determinada columna espacial.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                      110 140 22 3,
                                      120 130 26 4,
                                      110 120 20 3))', 0) )

INSERT INTO sample_polys
  VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                      0 4 35 9,
                                      5 4 32 12,
                                      5 0 31 5,
                                      0 0 40 7))', 0) )

INSERT INTO sample_polys
  VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                      8 4 10 12,
                                      9 4 12 11,
                                      12 13 10 16))', 0) )
```

Ejemplo 1:

Este ejemplo determina la coordenada X máxima de cada polígono contenido en SAMPLE_POLYS.

```
SELECT id, CAST ( ST_MaxX(geometry) AS INTEGER) MAX_X_COORD
  FROM sample_polys
```

Resultados:

```
ID          MAX_X_COORD
----------- ------------
          1          120
          2            5
          3           12
```

Ejemplo 2:

Este ejemplo determina la coordenada X máxima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxX(geometry) ) AS INTEGER) OVERALL_MAX_X
  FROM sample_polys
```

Resultados:

```
OVERALL_MAX_X
-------------
          120
```

Ejemplo 3:

Este ejemplo determina el rango espacial (diferencia entre el valor mínimo total y el valor máximo total) de todos los polígonos contenidos en la tabla SAMPLE_POLYS. Este cálculo se suele utilizar para comparar el rango espacial actual de las geometrías con el rango espacial del sistema de referencia espacial de los datos, para determinar si existe espacio para que el crecimiento de los datos.

```
SELECT CAST ( MIN (ST_MinX (geometry)) AS INTEGER) MIN_X,
       CAST ( MIN (ST_MinY (geometry)) AS INTEGER) MIN_Y,
       CAST ( MIN (ST_MinZ (geometry)) AS INTEGER) MIN_Z,
       CAST ( MIN (ST_MinM (geometry)) AS INTEGER) MIN_M,
       CAST ( MAX (ST_MaxX (geometry)) AS INTEGER) MAX_X,
       CAST ( MAX (ST_MaxY (geometry)) AS INTEGER) MAX_Y,
       CAST ( MAX (ST_MaxZ (geometry)) AS INTEGER) MAX_Z,
       CAST ( MAX (ST_MaxmM(geometry)) AS INTEGER) MAX_M,
  FROM sample_polys
```

Resultados:

```
MIN_X     MIN_Y     MIN_Z     MIN_M     MAX_X     MAX_Y     MAX_Z     MAX_M
--------- --------- --------- --------- --------- --------- --------- ----------
        0         0        10         3       120       140        40         16
```

Información relacionada:

- "ST_MaxM" en la página 436
- "ST_MaxY" en la página 439
- "ST_MaxZ" en la página 440
- "ST_MinX" en la página 449

ST_MaxY

ST_MaxY toma una geometría como parámetro de entrada y devuelve su coordenada Y máxima.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos para el que se devuelve la coordenada Y máxima.

Tipo de retorno:

DOUBLE

Ejemplos:

Estos ejemplos muestran el uso de la función ST_MaxY. Se crean tres polígonos y luego se insertan en la tabla SAMPLE_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)
```

```
INSERT INTO sample_polys
  VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                      110 140 22 3,
                                      120 130 26 4,
                                      110 120 20 3))', 0) )

INSERT INTO sample_polys
  VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                      0 4 35 9,
                                      5 4 32 12,
                                      5 0 31 5,
                                      0 0 40 7))', 0) )

INSERT INTO sample_polys
  VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                      8 4 10 12,
                                      9 4 12 11,
                                      12 13 10 16))', 0) )
```

Ejemplo 1:

Este ejemplo determina la coordenada Y máxima de cada polígono contenido en SAMPLE_POLYS.

```
SELECT id, CAST ( ST_MaxY(geometry) AS INTEGER) MAX_Y
  FROM sample_polys
```

Resultados:

```
ID          MAX_Y
----------- ------------
          1          140
          2            4
          3           13
```

Ejemplo 2:

Este ejemplo determina la coordenada Y máxima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxY(geometry) ) AS INTEGER) OVERALL_MAX_Y
  FROM sample_polys
```

Resultados:

```
OVERALL_MAX_Y
--------------
           140
```

Conceptos relacionados:

- "ST_MaxX" en la página 437

Información relacionada:

- "ST_MaxM" en la página 436
- "ST_MaxZ" en la página 440
- "ST_MinY" en la página 450

ST_MaxZ

ST_MaxZ toma una geometría como parámetro de entrada y devuelve su coordenada Z máxima.

Si la geometría proporcionada es un valor nulo o está vacía, o si no tiene coordenadas Z, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos para el que se devuelve la coordenada Z máxima.

Tipo de retorno:

DOUBLE

Ejemplos:

Estos ejemplos muestran el uso de la función ST_MaxZ. Se crean tres polígonos y luego se insertan en la tabla SAMPLE_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                      110 140 22 3,
                                      120 130 26 4,
                                      110 120 20 3))', 0) )

INSERT INTO sample_polys
  VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                      0 4 35 9,
                                      5 4 32 12,
                                      5 0 31 5,
                                      0 0 40 7))', 0) )

INSERT INTO sample_polys
  VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                      8 4 10 12,
                                      9 4 12 11,
                                      12 13 10 16))', 0) )
```

Ejemplo 1:

Este ejemplo determina la coordenada Z máxima de cada polígono contenido en SAMPLE_POLYS.

```
SELECT id, CAST ( ST_MaxZ(geometry) AS INTEGER) MAX_Z
  FROM sample_polys
```

Resultados:

```
ID          MAX_Z
----------- -----------
          1          26
          2          40
          3          12
```

Ejemplo 2:

Este ejemplo determina la coordenada Z máxima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxZ(geometry) ) AS INTEGER) OVERALL_MAX_Z
  FROM sample_polys
```

Resultados:

```
OVERALL_MAX_Z
-------------
           40
```

Conceptos relacionados:

- "ST_MaxX" en la página 437

Información relacionada:

- "ST_MaxM" en la página 436
- "ST_MaxY" en la página 439
- "ST_MinZ" en la página 452

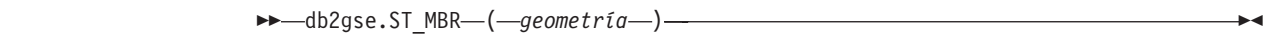
ST_MBR

ST_MBR toma una geometría como parámetro de entrada y devuelve el rectángulo delimitador mínimo de la misma.

Si dicha geometría es un punto, se devuelve el propio punto. Si la geometría es una cadena lineal horizontal o vertical y el sistema de referencia espacial no es geodésico, se devuelve la propia cadena lineal horizontal o vertical. En otro caso, se devuelve el rectángulo delimitador mínimo de la geometría en forma de polígono. Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la que se obtiene el rectángulo delimitador mínimo.

Tipo de retorno:

db2gse.ST_Geometry

Ejemplo:

Este ejemplo muestra cómo utilizar la función ST_MBR para obtener el rectángulo delimitador mínimo de un polígono. Debido a que la geometría especificada es un polígono, el rectángulo delimitador mínimo se devuelve en forma de polígono.

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon ('polygon (( 5 5, 7 7, 5 9, 7 9, 9 11, 13 9,
                                     15 9, 13 7, 15 5, 9 6, 5 5))', 0) )

INSERT INTO sample_polys
  VALUES (2, ST_Polygon ('polygon (( 20 30, 25 35, 30 30, 20 30))', 0) )

SELECT id, CAST (ST_AsText ( ST_MBR(geometry)) AS VARCHAR(150) ) MBR
  FROM sample_polys
```

Resultados:

```
ID        MBR
--------- ------------------------------------------------------------
        1 POLYGON (( 5.00000000 5.00000000, 15.00000000 5.00000000,
             15.00000000 11.00000000, 5.00000000 11.00000000,
              5.00000000 5.00000000))
        2 POLYGON (( 20.00000000 30.00000000, 30.00000000 30.00000000,
             30.00000000 35.00000000, 20.00000000 35.00000000,
             20.00000000 30.00000000 ))
```

Información relacionada:

- "ST_Envelope" en la página 390
- "ST_MBRIntersects" en la página 443

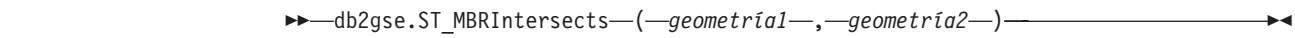
ST_MBRIntersects

ST_MBRIntersects toma dos geometrías como parámetros de entrada y devuelve un 1 si los rectángulos delimitadores mínimos de las dos geometrías forman intersección. En caso contrario, se devuelve un 0 (cero). El rectángulo delimitador mínimo de un punto y de una cadena lineal horizontal o vertical es la propia geometría.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si cualquiera de las dos geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.

Sintaxis:



Parámetros:

geometría1
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría cuyo rectángulo delimitador mínimo se debe verificar si forma intersección con el rectángulo delimitador mínimo de *geometría2*.

geometría2
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría cuyo rectángulo delimitador mínimo se debe verificar si forma intersección con el rectángulo delimitador mínimo de *geometría1*.

Tipo de retorno:

INTEGER

Ejemplos:

Estos ejemplos muestran el uso de ST_MBRIntersects para obtener una aproximación de si dos polígonos inconexos están próximos entre sí observando si sus rectángulos delimitadores mínimos forman intersección. El primer ejemplo utiliza la expresión CASE de SQL. El segundo ejemplo utiliza una sentencia SELECT individual para encontrar los polígonos que forman intersección con el rectángulo delimitador mínimo del polígono cuyo ID es 2.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon ('polygon (( 0 0, 30 0, 40 30, 40 35,
                                     5 35, 5 10, 20 10, 20 5, 0 0 ))', 0) )

INSERT INTO sample_polys
  VALUES (2, ST_Polygon ('polygon (( 15 15, 15 20, 60 20, 60 15,
                                     15 15 ))', 0) )

INSERT INTO sample_polys
  VALUES (3, ST_Polygon ('polygon (( 115 15, 115 20, 160 20, 160 15,
                                     115 15 ))', 0) )
```

Ejemplo 1:

La sentencia SELECT siguiente utiliza una expresión CASE para encontrar los ID de los polígonos que tienen rectángulos delimitadores mínimos que forman intersección.

```
SELECT a.id, b.id,
    CASE ST_MBRIntersects (a.geometry, b.geometry)
      WHEN 0 THEN 'MBRs do not intersect'
      WHEN 1 THEN 'MBRs intersect'
    END AS MBR_INTERSECTS
  FROM sample_polys a, sample_polys b
  WHERE a.id <= b.id
```

Resultados:

```
ID         ID          MBR_INTERSECTS
---------- ---------- --------------
         1          1 MBRs intersect
         1          2 MBRs intersect
         2          2 MBRs intersect
         1          3 MBRs do not intersect
         2          3 MBRs do not intersect
         3          3 MBRs intersect
```

Ejemplo 2:

La sentencia SELECT siguiente determina si los rectángulos delimitadores mínimos de las geometrías forman intersección con el del polígono cuyo ID es 2.

```
SELECT a.id, b.id, ST_MBRIntersects (a.geometry, b.geometry) MBR_INTERSECTS
  FROM sample_polys a, sample_polys b
  WHERE a.id = 2
```

Resultados

```
ID          ID           MBR_INTERSECTS
----------- ------------ ---------------
          2            1               1
          2            2               1
          2            3               0
```

Información relacionada:

- "ST_EnvIntersects" en la página 392
- "ST_MBR" en la página 442

ST_MeasureBetween, ST_LocateBetween

ST_MeasureBetween o ST_LocateBetween toma como parámetros de entrada una geometría y dos coordenadas M (medidas) y devuelve la parte de la geometría proporcionada que representa el conjunto de puntos desconectados situados entre las dos coordenadas M.

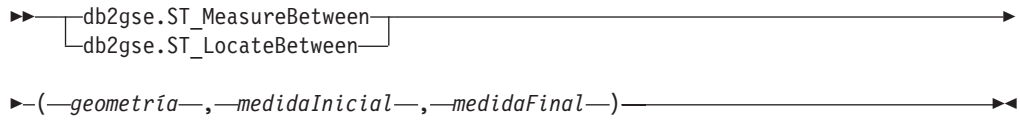
En el caso de curvas, multicurvas, superficies y multisuperficies, se realiza una interpolación para calcular el resultado. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si la geometría proporcionada es una superficie o multisuperficie, ST_MeasureBetween o ST_LocateBetween se aplica a los anillos exterior e interior de la geometría. Si ninguna de las partes de la geometría proporcionada está dentro del intervalo definido por las coordenadas M proporcionadas, el resultado es una geometría vacía. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

La geometría resultante se representa según el tipo espacial más apropiado. Si se puede representar como punto, cadena lineal o polígono, se utiliza uno de estos tipos. En otro caso, se representa como multipunto, multilínea o multipolígono.

También se puede invocar a ambas funciones como métodos.

Sintaxis:



Parámetros:

geometría
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría donde se deben localizar las partes cuyos valores de medida están comprendidos entre *medidaInicial* y *medidaFinal*.

medidaInicial
: Valor de tipo DOUBLE que representa el límite inferior del intervalo de medidas. Si este valor es nulo, no se aplica ningún límite inferior.

medidaFinal
: Valor de tipo DOUBLE que representa el límite superior del intervalo de medidas. Si este valor es nulo, no se aplica ningún límite superior.

Tipo de retorno:

db2gse.ST_Geometry

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

La coordenada M (medida) de una geometría está definida por el usuario. Es muy versátil porque puede representar cualquier entidad que desee medir; por ejemplo, una distancia de carretera, o medidas de temperatura, presión o pH.

Este ejemplo muestra el uso de la coordenada M para registrar datos recogidos de mediciones de pH. Un investigador obtiene la medida del pH del suelo en lugares determinados de una carretera. De acuerdo con sus técnicas habituales de trabajo, anota los valores necesarios en cada lugar donde toma una muestra del suelo: las coordenadas X e Y de ese lugar y el pH medido.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, geometry ST_LineString)

INSERT INTO sample_lines
  VALUES (1, ST_LineString ('linestring m (2 2 3, 3 5 3,
                                           3 3 6, 4 4 6,
                                           5 5 6, 6 6 8)', 1 ) )
```

Para encontrar el tramo donde la acidez del suelo oscila entre 4 y 6, el investigador utilizaría esta sentencia SELECT:

```
SELECT id, CAST( ST_AsText( ST_MeasureBetween( 4, 6) )
  AS VARCHAR(150) ) MEAS_BETWEEN_4_AND_6
  FROM sample_lines
```

Resultados:

```
ID         MEAS_BETWEEN_4_AND_6
---------- ------------------------------------------------------
         1 LINESTRING M (3.00000000 4.33333300 4.00000000,
                         3.00000000 3.00000000 6.00000000,
                         4.00000000 4.00000000 6.00000000,
                         5.00000000 5.00000000 6.00000000)
```

ST_MidPoint

ST_MidPoint toma una curva como parámetro de entrada y devuelve un punto de la curva que es equidistante de los dos puntos finales de la curva, medido a lo largo de la curva. El punto resultante se representa según el sistema de referencia espacial de la curva proporcionada.

Si la curva proporcionada está vacía, se devuelve un punto vacío. Si la curva proporcionada es un valor nulo, se devuelve un valor nulo.

Si la curva contiene coordenadas Z o coordenadas M (medidas), el punto medio se determina basándose solamente en los valores de las coordenadas Y e Y de la curva. La coordenada Z y la medida del punto obtenido se interpolan.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

curva Valor de tipo ST_Curve o uno de sus subtipos que representa la curva para la que se obtiene el punto medio.

Tipo de retorno:

db2gse.ST_Point

Ejemplo:

Este ejemplo muestra el uso de ST_MidPoint para obtener el punto medio de curvas.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, geometry ST_LineString)

INSERT INTO sample_lines (id, geometry)
  VALUES (1, ST_LineString ('linestring (0 0, 0 10, 0 20, 0 30, 0 40)', 1 ) )

INSERT INTO sample_lines (id, geometry)
  VALUES (2, ST_LineString ('linestring (2 2, 3 5, 3 3, 4 4, 5 5, 6 6)', 1 ) )

INSERT INTO sample_lines (id, geometry)
  VALUES (3, ST_LineString ('linestring (0 10, 0 0, 10 0, 10 10)', 1 ) )

INSERT INTO sample_lines (id, geometry)
  VALUES (4, ST_LineString ('linestring (0 20, 5 20, 10 20, 15 20)', 1 ) )

SELECT id, CAST( ST_AsText( ST_MidPoint(geometry) ) AS VARCHAR(60) ) MID_POINT
  FROM sample_lines
```

Resultados:

```
ID         MID_POINT
---------- ------------------------------------
         1 POINT ( 0.00000000 20.00000000)
         2 POINT ( 3.00000000 3.45981800)
         3 POINT ( 5.00000000 0.00000000)
         4 POINT ( 7.50000000 20.00000000)
```

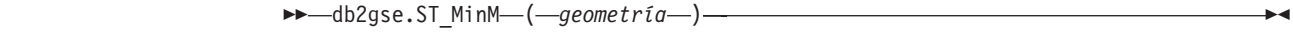
ST_MinM

ST_MinM toma una geometría como parámetro de entrada y devuelve su coordenada M mínima.

Si la geometría proporcionada es un valor nulo o está vacía, o si no tiene coordenadas M, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos para el que se devuelve la coordenada M mínima.

Tipo de retorno:

DOUBLE

Ejemplos:

Estos ejemplos muestran el uso de la función ST_MinM. Se crean tres polígonos y luego se insertan en la tabla SAMPLE_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                      110 140 22 3,
                                      120 130 26 4,
                                      110 120 20 3))', 0) )

INSERT INTO sample_polys
  VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                      0 4 35 9,
                                      5 4 32 12,
                                      5 0 31 5,
                                      0 0 40 7))', 0) )

INSERT INTO sample_polys
  VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                      8 4 10 12,
                                      9 4 12 11,
                                      12 13 10 16))', 0) )
```

Ejemplo 1:

Este ejemplo determina la coordenada M mínima de cada polígono contenido en SAMPLE_POLYS.

```
SELECT id, CAST ( ST_MinM(geometry) AS INTEGER) MIN_M
  FROM sample_polys
```

Resultados:

```
ID          MIN_M
----------- -----------
          1           3
          2           5
          3          11
```

Ejemplo 2:

Este ejemplo determina la coordenada M mínima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MIN ( ST_MinM(geometry) ) AS INTEGER) OVERALL_MIN_M
  FROM sample_polys
```

Resultados:

```
OVERALL_MIN_M
--------------
3
```

Información relacionada:

- "ST_MaxM" en la página 436
- "ST_MinX" en la página 449
- "ST_MinY" en la página 450
- "ST_MinZ" en la página 452

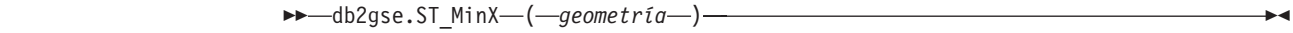
ST_MinX

ST_MinX toma una geometría como parámetro de entrada y devuelve la coordenada X mínima.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos para el que se devuelve la coordenada X mínima.

Tipo de retorno:

DOUBLE

Ejemplos:

Estos ejemplos muestran el uso de la función ST_MinX. Se crean tres polígonos y luego se insertan en la tabla SAMPLE_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                      110 140 22 3,
                                      120 130 26 4,
                                      110 120 20 3))', 0) )

INSERT INTO sample_polys
  VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                      0 4 35 9,
                                      5 4 32 12,
                                      5 0 31 5,
                                      0 0 40 7))', 0) )

INSERT INTO sample_polys
  VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                      8 4 10 12,
                                      9 4 12 11,
                                      12 13 10 16))', 0) )
```

Ejemplo 1:

Este ejemplo determina la coordenada X mínima de cada polígono contenido en SAMPLE_POLYS.

```
SELECT id, CAST ( ST_MinX(geometry) AS INTEGER) MIN_X
  FROM sample_polys
```

Resultados:

```
ID          MIN_X
----------- ------------
          1          110
          2            0
          3            8
```

Ejemplo 2:

Este ejemplo determina la coordenada X mínima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MIN ( ST_MinX(geometry) ) AS INTEGER) OVERALL_MIN_X
  FROM sample_polys
```

Resultados:

```
OVERALL_MIN_X
-------------
            0
```

Conceptos relacionados:

- "ST_MaxX" en la página 437

Información relacionada:

- "ST_MinM" en la página 447
- "ST_MinY" en la página 450
- "ST_MinZ" en la página 452

ST_MinY

ST_MinY toma una geometría como parámetro de entrada y devuelve su coordenada Y mínima.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
: Valor de tipo ST_Geometry o uno de sus subtipos para el que se devuelve la coordenada Y mínima.

Tipo de retorno:

DOUBLE

Ejemplos:

Estos ejemplos muestran el uso de la función ST_MinY. Se crean tres polígonos y luego se insertan en la tabla SAMPLE_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                      110 140 22 3,
                                      120 130 26 4,
                                      110 120 20 3))', 0) )

INSERT INTO sample_polys
  VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                      0 4 35 9,
                                      5 4 32 12,
                                      5 0 31 5,
                                      0 0 40 7))', 0) )

INSERT INTO sample_polys
  VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                      8 4 10 12,
                                      9 4 12 11,
                                      12 13 10 16))', 0) )
```

Ejemplo 1:

Este ejemplo determina la coordenada Y mínima de cada polígono contenido en SAMPLE_POLYS.

```
SELECT id, CAST ( ST_MinY(geometry) AS INTEGER) MIN_Y
  FROM sample_polys
```

Resultados:

```
ID          MIN_Y
----------- ------------
          1          120
          2            0
          3            4
```

Ejemplo 2:

Este ejemplo determina la coordenada Y mínima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MIN ( ST_MinY(geometry) ) AS INTEGER) OVERALL_MIN_Y
  FROM sample_polys
```

Resultados:

```
OVERALL_MIN_Y
--------------
            0
```

Información relacionada:

- “ST_MaxY” en la página 439
- “ST_MinM” en la página 447
- “ST_MinX” en la página 449
- “ST_MinZ” en la página 452

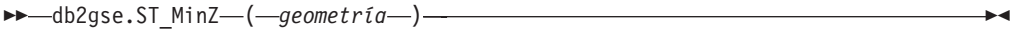
ST_MinZ

ST_MinZ toma una geometría como parámetro de entrada y devuelve su coordenada Z mínima.

Si la geometría proporcionada es un valor nulo o está vacía, o si no tiene coordenadas Z, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos para el que se devuelve la coordenada Z mínima.

Tipo de retorno:

DOUBLE

Ejemplos:

Estos ejemplos muestran el uso de la función ST_MinZ. Se crean tres polígonos y luego se insertan en la tabla SAMPLE_POLYS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon('polygon zm ((110 120 20 3,
                                      110 140 22 3,
                                      120 130 26 4,
                                      110 120 20 3))', 0) )

INSERT INTO sample_polys
  VALUES (2, ST_Polygon('polygon zm ((0 0 40 7,
                                      0 4 35 9,
                                      5 4 32 12,
                                      5 0 31 5,
                                      0 0 40 7))', 0) )

INSERT INTO sample_polys
  VALUES (3, ST_Polygon('polygon zm ((12 13 10 16,
                                       8 4 10 12,
                                       9 4 12 11,
                                      12 13 10 16))', 0) )
```

Ejemplo 1:

Este ejemplo determina la coordenada Z mínima de cada polígono contenido en SAMPLE_POLYS.

```
SELECT id, CAST ( ST_MinZ(geometry) AS INTEGER) MIN_Z
  FROM sample_polys
```

Resultados:

```
ID          MIN_Z
----------- ------------
          1           20
          2           31
          3           10
```

Ejemplo 2:

Este ejemplo determina la coordenada Z mínima existente para todos los polígonos de la columna GEOMETRY.

```
SELECT CAST ( MIN ( ST_MinZ(geometry) ) AS INTEGER) OVERALL_MIN_Z
  FROM sample_polys
```

Resultados:

```
OVERALL_MIN_Z
-------------
10
```

Información relacionada:

- "ST_MaxZ" en la página 440
- "ST_MinM" en la página 447
- "ST_MinX" en la página 449
- "ST_MinY" en la página 450

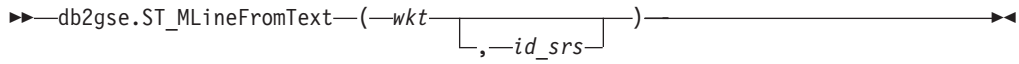
ST_MLineFromText

ST_MLineFromText toma como parámetros de entrada una representación de texto convencional de una multilínea y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la multilínea correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST_MultiLineString. Es recomendable debido a su flexibilidad: ST_MultiLineString acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

Sintaxis:



Parámetros:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la multilínea resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la multilínea resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_MultiLineString

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST_MLineFromText para crear e insertar una multilínea a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es una multilínea representada en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son:

- Línea 1: (33, 2) (34, 3) (35, 6)
- Línea 2: (28, 4) (29, 5) (31, 8) (43, 12)
- Línea 3: (39, 3) (37, 4) (36, 7)

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mlines (id INTEGER, geometry ST_MultiLineString)

INSERT INTO sample_mlines
  VALUES (1110, ST_MLineFromText ('multilinestring ( (33 2, 34 3, 35 6),
                                                     (28 4, 29 5, 31 8, 43 12),
                                                     (39 3, 37 4, 36 7) )', 1) )
```

La sentencia SELECT siguiente devuelve la multilínea que se registró en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(280) ) MULTI_LINE_STRING
  FROM sample_mlines
  WHERE id = 1110
```

Resultados:

```
ID         MULTI_LINE_STRING
---------- ------------------------------------------------------------------
      1110 MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000 3.00000000,
              35.00000000 6.00000000),
                           ( 28.00000000 4.00000000, 29.00000000 5.00000000,
              31.00000000 8.0000000, 43.00000000 12.00000000),
                           ( 39.00000000 3.00000000, 37.00000000 4.00000000,
              36.00000000 7.00000000 ))
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_MLineFromWKB" en la página 454
- "ST_MultiLineString" en la página 462

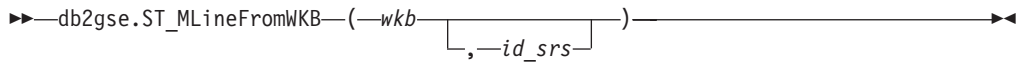
ST_MLineFromWKB

ST_MLineFromWKB toma como parámetros de entrada una representación binaria convencional de una multilínea y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve la multilínea correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST_MultiLineString. Es recomendable debido a su flexibilidad: ST_MultiLineString acepta otras modalidades de datos de entrada, además de la representación binaria convencional.

Sintaxis:



Parámetros:

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la multilínea resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la multilínea resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_MultiLineString

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST_MLineFromWKB para crear e insertar una multilínea a partir de su representación binaria convencional. La geometría es una multilínea en el sistema de referencia espacial 1. En este ejemplo, la multilínea se guarda con el ID = 10 en la columna GEOMETRY de la tabla SAMPLE_MLINES, y luego la columna WKB se actualiza con su representación binaria convencional (utilizando la función ST_AsBinary). Finalmente, se utiliza la función ST_MLineFromWKB para obtener la multilínea a partir de la columna WKB. Las coordenadas X e Y de esta geometría son:

- Línea 1: (61, 2) (64, 3) (65, 6)
- Línea 2: (58, 4) (59, 5) (61, 8)
- Línea 3: (69, 3) (67, 4) (66, 7) (68, 9)

La tabla SAMPLE_MLINES tiene una columna GEOMETRY, donde se guarda la multilínea, y una columna WKB, donde se guarda la representación binaria convencional de la multilínea.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mlines (id INTEGER, geometry ST_MultiLineString,
       wkb BLOB(32K))

INSERT INTO sample_mlines
  VALUES (10, ST_MultiLineString ('multilinestring
       ( (61 2, 64 3, 65 6),
         (58 4, 59 5, 61 8),
```

```
        (69 3, 67 4, 66 7, 68 9) )', 1) )

UPDATE sample_mlines AS temporary_correlated
  SET wkb = ST_AsBinary( geometry )
  WHERE id = temporary_correlated.id
```

En la sentencia SELECT siguiente, se utiliza la función ST_MLineFromWKB para obtener la multilínea a partir de la columna WKB.

```
SELECT id, CAST( ST_AsText( ST_MLineFromWKB (wkb) )
       AS VARCHAR(280) ) MULTI_LINE_STRING
  FROM sample_mlines
  WHERE id = 10
```

Resultados:

```
ID         MULTI_LINE_STRING
---------- ------------------------------------------------------------------
        10 MULTILINESTRING (( 61.00000000 2.00000000, 64.00000000 3.00000000,
              65.00000000 6.00000000),
                           ( 58.00000000 4.00000000, 59.00000000 5.00000000,
              61.00000000 8.0000000),
                           ( 69.00000000 3.00000000, 67.00000000 4.00000000,
              66.00000000 7.00000000, 68.00000000 9.00000000 ))
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_MLineFromText" en la página 453
- "ST_MultiLineString" en la página 462
- "Representación binaria convencional (WKB)" en la página 534

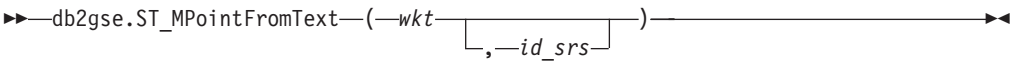
ST_MPointFromText

ST_MPointFromText toma como parámetros de entrada una representación de texto convencional de un multipunto y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el multipunto correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST_MultiPoint. Es recomendable debido a su flexibilidad: ST_MultiPoint acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

Sintaxis:



Parámetros:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional del multipunto resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipunto resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_MultiPoint

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST_MPointFromText para crear e insertar una multipunto a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es un multipunto representada en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son: (1, 2) (4, 3) (5, 6).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpoints (id INTEGER, geometry ST_MultiPoint)

INSERT INTO sample_mpoints
  VALUES (1110, ST_MPointFromText ('multipoint (1 2, 4 3, 5 6) )', 1) )
```

La sentencia SELECT siguiente devuelve el multipunto que se registró en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(280) ) MULTIPOINT
  FROM sample_mpoints
  WHERE id = 1110
```

Resultados:

```
ID         MULTIPOINT
---------- ----------------------------------------------------------------------
      1110 MULTIPOINT (1.00000000 2.00000000, 4.00000000 3.00000000,
                       5.00000000 6.00000000)
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_MPointFromWKB" en la página 457
- "ST_MultiPoint" en la página 464
- "Representación de texto convencional (WKT)" en la página 529

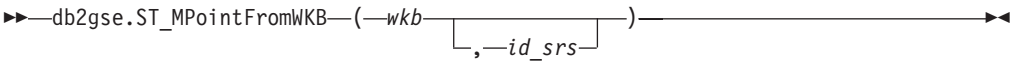
ST_MPointFromWKB

ST_MPointFromWKB toma como parámetros de entrada una representación binaria convencional de un multipunto y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el multipunto correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST_MultiPoint. Es recomendable debido a su flexibilidad: ST_MultiPoint acepta otras modalidades de datos de entrada, además de la representación binaria convencional.

Sintaxis:



Parámetros:

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional del multipunto resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipunto resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_MultiPoint

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST_MPointFromWKB para crear e insertar un multipunto a partir de su representación binaria convencional. La geometría es una multipunto en el sistema de referencia espacial 1. En este ejemplo, el multipunto se guarda con el ID = 10 en la columna GEOMETRY de la tabla SAMPLE_MPOINTS, y luego la columna WKB se actualiza con su representación binaria convencional (utilizando la función ST_AsBinary). Finalmente, se utiliza la función ST_MPointFromWKB para obtener el multipunto a partir de la columna WKB. Las coordenadas X e Y de esta geometría son: (44, 14) (35, 16) (24, 13).

La tabla SAMPLE_MPOINTS tiene una columna GEOMETRY, donde se guarda el multipunto, y una columna WKB, donde se guarda la representación binaria convencional del multipunto.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpoints (id INTEGER, geometry ST_MultiPoint,
    wkb BLOB(32K))

INSERT INTO sample_mpoints
  VALUES (10, ST_MultiPoint ('multipoint ( 4 14, 35 16, 24 13)', 1))

UPDATE sample_mpoints AS temporary_correlated
  SET wkb = ST_AsBinary( geometry )
  WHERE id = temporary_correlated.id
```

En la sentencia SELECT siguiente, se utiliza la función ST_MPointFromWKB para obtener el multipunto a partir de la columna WKB.

```
SELECT id, CAST( ST_AsText( ST_MLineFromWKB (wkb)) AS VARCHAR(100)) MULTIPOINT
  FROM sample_mpoints
  WHERE id = 10
```

Resultados:

```
ID         MULTIPOINT
---------- ----------------------------------------------------------------
        10 MULTIPOINT (44.00000000 14.00000000, 35.00000000
             16.00000000 24.00000000 13.00000000)
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_MPointFromText" en la página 456
- "ST_MultiPoint" en la página 464
- "Representación binaria convencional (WKB)" en la página 534
- "ST_Point" en la página 478

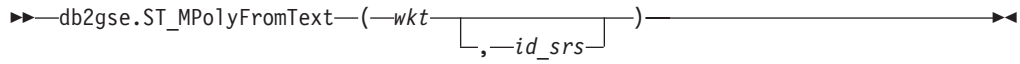
ST_MPolyFromText

ST_MPolyFromText toma como parámetros de entrada una representación de texto convencional de un multipolígono y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el multipolígono correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST_MultiPolygon. Es recomendable debido a su flexibilidad: ST_MultiPolygon acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

Sintaxis:



Parámetros:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional del multipolígono resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipolígono resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_MultiPolygon

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST_MPolyFromText para crear e insertar un multipolígono a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es un multipolígono representado en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son:

- Polígono 1: (3, 3) (4, 6) (5, 3) (3, 3)
- Polígono 2: (8, 24) (9, 25) (1, 28) (8, 24)
- Polígono 3: (13, 33) (7, 36) (1, 40) (10, 43) (13, 33)

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER, geometry ST_MultiPolygon)

INSERT INTO sample_mpolys
  VALUES (1110,
          ST_MPolyFromText ('multipolygon (( (3 3, 4 6, 5 3, 3 3),
                                (8 24, 9 25, 1 28, 8 24),
                                (13 33, 7 36, 1 40, 10 43 13 33) ))', 1) )
```

La sentencia SELECT siguiente devuelve la multipolígono que se registró en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(350) ) MULTI_POLYGON
  FROM sample_mpolys
  WHERE id = 1110
```

Resultados:

```
ID          MULTI_POLYGON
------- ------------------------------------------------------------------
   1110 MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000,
           1.00000000 40.00000000, 7.00000000 36.00000000,
           13.00000000 33.00000000)),
                      (( 8.00000000 24.00000000, 9.00000000 25.00000000,
           1.00000000 28.0000000, 8.00000000 24.00000000)),
                       ( 3.00000000 3.00000000, 5.00000000 3.00000000,
           4.00000000 6.00000000, 3.00000000 3.00000000)))
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_MPolyFromWKB" en la página 460
- "ST_MultiPolygon" en la página 465
- "Representación de texto convencional (WKT)" en la página 529

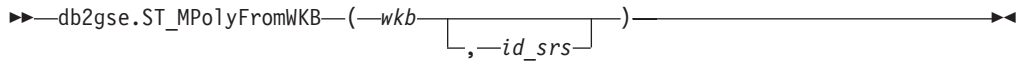
ST_MPolyFromWKB

ST_MPolyFromWKB toma como parámetros de entrada una representación binaria convencional de un multipolígono y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el multipolígono correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST_MultiPolygon. Es recomendable debido a su flexibilidad: ST_MultiPolygon acepta otras modalidades de datos de entrada, además de la representación binaria convencional.

Sintaxis:



Parámetros:

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional del multipolígono resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipolígono resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial contenidos en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_MultiPolygon

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST_MPolyFromWKB para crear un multipolígono a partir de su representación binaria convencional. La geometría es un multipolígono en el sistema de referencia espacial 1. En este ejemplo, el multipolígono se guarda con el ID = 10 en la columna GEOMETRY de la tabla SAMPLE_MPOLYS, y luego la columna WKB se actualiza con su representación binaria convencional (utilizando la función ST_AsBinary). Finalmente, se utiliza la función ST_MPolyFromWKB para obtener el multipolígono a partir de la columna WKB. Las coordenadas X e Y de esta geometría son:

- Polígono 1: (1, 72) (4, 79) (5, 76) (1, 72)
- Polígono 2: (10, 20) (10, 40) (30, 41) (10, 20)
- Polígono 3: (9, 43) (7, 44) (6, 47) (9, 43)

La tabla SAMPLE_MPOLYS tiene una columna GEOMETRY, donde se guarda el multipolígono, y una columna WKB, donde se guarda la representación binaria convencional del multipolígono.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER,
    geometry ST_MultiPolygon, wkb BLOB(32K))

INSERT INTO sample_mpolys
  VALUES (10, ST_MultiPolygon ('multipolygon
        (( (1 72, 4 79, 5 76, 1 72),
         (10 20, 10 40, 30 41, 10 20),
         (9 43, 7 44, 6 47, 9 43) ))', 1))
```

```
UPDATE sample_mpolys AS temporary_correlated
  SET wkb = ST_AsBinary( geometry )
  WHERE id = temporary_correlated.id
```

En la sentencia SELECT siguiente, se utiliza la función ST_MPolyFromWKB para obtener el multipolígono a partir de la columna WKB.

```
SELECT id, CAST( ST_AsText( ST_MPolyFromWKB (wkb) )
    AS VARCHAR(320) ) MULTIPOLYGON
  FROM sample_mpolys
  WHERE id = 10
```

Resultados:

```
ID         MULTIPOLYGON
---------- ------------------------------------------------------------------
        10 MULTIPOLYGON ((( 10.00000000 20.00000000, 30.00000000
              41.00000000, 10.00000000 40.00000000, 10.00000000
              20.00000000)),
                         ( 1.00000000 72.00000000, 5.00000000
              76.00000000, 4.00000000 79.0000000, 1.00000000
              72,00000000)),
                          ( 9.00000000 43.00000000, 6.00000000
               47.00000000, 7.00000000 44.00000000, 9.00000000
               43.00000000 )))
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_MPolyFromText" en la página 459
- "ST_MultiPolygon" en la página 465
- "Representación binaria convencional (WKB)" en la página 534
- "ST_Polygon" en la página 488

ST_MultiLineString

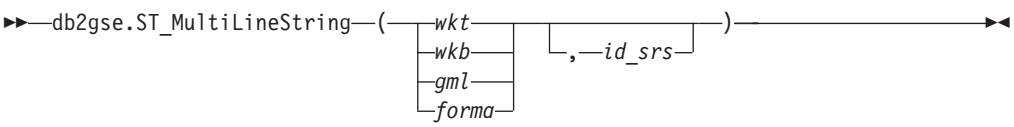
ST_MultiLineString crea una multilínea a partir de uno de estos datos de entrada:

- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenida la multilínea resultante.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma o la representación GML es nula, se devuelve un valor nulo.

Sintaxis:



Parámetros:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la multilínea resultante.

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la multilínea resultante.

gml Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), de la multilínea resultante.

forma Valor de tipo BLOB(2G) que contiene la representación de forma de la multilínea resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la multilínea resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_MultiLineString

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST_MultiLineString para crear e insertar una multilínea a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es una multilínea representada en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son:

- Línea 1: (33, 2) (34, 3) (35, 6)
- Línea 2: (28, 4) (29, 5) (31, 8) (43, 12)
- Línea 3: (39, 3) (37, 4) (36, 7)

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mlines (id INTEGER,
                            geometry ST_MultiLineString)

INSERT INTO sample_mlines
  VALUES (1110,
          ST_MultiLineString ('multilinestring ( (33 2, 34 3, 35 6),
                                      (28 4, 29 5, 31 8, 43 12),
                                      (39 3, 37 4, 36 7) )', 1) )
```

La sentencia SELECT siguiente devuelve la multilínea que se registró en la tabla:

```
SELECT id,
       CAST( ST_AsText( geometry ) AS VARCHAR(280) )
  MULTI_LINE_STRING
  FROM sample_mlines
  WHERE id = 1110
```

Resultados:

```
ID         MULTI_LINE_STRING
------- ------------------------------------------------------------------
   1110 MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000 3.00000000,
              35.00000000 6.00000000),
                         ( 28.00000000 4.00000000, 29.00000000 5.00000000,
              31.00000000 8.0000000, 43.00000000 12.00000000),
                         ( 39.00000000 3.00000000, 37.00000000 4.00000000,
              36.00000000 7.00000000 ))
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "Representación de texto convencional (WKT)" en la página 529
- "Representación binaria convencional (WKB)" en la página 534
- "Representación de forma" en la página 536
- "Representación GML (Geography Markup Language)" en la página 536

ST_MultiPoint

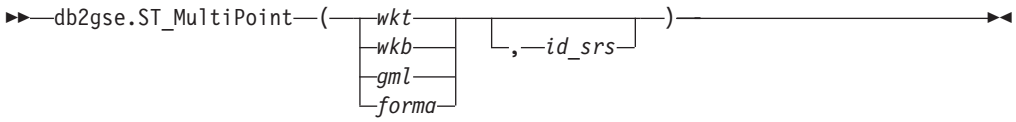
ST_MultiPoint crea un multipunto a partir de uno de estos datos de entrada:

- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenido el multipunto resultante.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma o la representación GML es nula, se devuelve un valor nulo.

Sintaxis:



Parámetros:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional del multipunto resultante.

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional del multipunto resultante.

gml Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), del multipunto resultante.

forma Valor de tipo BLOB(2G) que contiene la representación de forma del multipunto resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipunto resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_Point

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST_MultiPoint para crear e insertar un multipunto a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es un multipunto representada en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son: (1, 2) (4, 3) (5, 6).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpoints (id INTEGER, geometry ST_MultiPoint)

INSERT INTO sample_mpoints
  VALUES (1110, ST_MultiPoint ('multipoint (1 2, 4 3, 5 6) )', 1))
```

La sentencia SELECT siguiente devuelve el multipunto que se registró en la tabla:

```
SELECT id, CAST( ST_AsText(geometry) AS VARCHAR(90)) MULTIPOINT
  FROM sample_mpoints
  WHERE id = 1110
```

Resultados:

```
ID         MULTIPOINT
---------- ----------------------------------------------------
      1110 MULTIPOINT (1.00000000 2.00000000, 4.00000000
           3.00000000, 5.00000000 6.00000000)
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "Representación de texto convencional (WKT)" en la página 529
- "Representación binaria convencional (WKB)" en la página 534
- "Representación de forma" en la página 536
- "Representación GML (Geography Markup Language)" en la página 536

ST_MultiPolygon

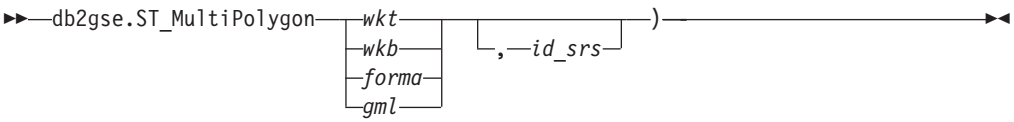
ST_MultiPolygon crea un multipolígono a partir de uno de estos datos de entrada:

- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenido el multipolígono resultante.

Si la representación de texto convencional, la representación binaria convencional, la representación de forma o la representación GML es nula, se devuelve un valor nulo.

Sintaxis:



Parámetros:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional del multipolígono resultante.

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional del multipolígono resultante.

gml Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), del multipolígono resultante.

forma Valor de tipo BLOB(2G) que contiene la representación de forma del multipolígono resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al multipolígono resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_MultiPolygon

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST_MultiPolygon para crear e insertar un multipolígono a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es un multipolígono representado en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son:

- Polígono 1: (3, 3) (4, 6) (5, 3) (3, 3)
- Polígono 2: (8, 24) (9, 25) (1, 28) (8, 24)
- Polígono 3: (13, 33) (7, 36) (1, 40) (10, 43) (13, 33)

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER, geometry ST_MultiPolygon)
```

```
INSERT INTO sample_mpolys
  VALUES (1110,
          ST_MultiPolygon ('multipolygon (( (3 3, 4 6, 5 3, 3 3),
                                    (8 24, 9 25, 1 28, 8 24),
                                    (13 33, 7 36, 1 40, 10 43 13 33) ))', 1) )
```

La sentencia SELECT siguiente devuelve la multipolígono que se registró en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(350) ) MULTI_POLYGON
  FROM sample_mpolys
  WHERE id = 1110
```

Resultados:

```
ID          MULTI_POLYGON
------- ------------------------------------------------------------------
   1110 MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000,
           1.00000000 40.00000000, 7.00000000 36.00000000,
          13.00000000 33.00000000)),
                      (( 8.00000000 24.00000000, 9.00000000 25.00000000,
           1.00000000 28.0000000, 8.00000000 24.00000000)),
                      (( 3.00000000 3.00000000, 5.00000000 3.00000000,
           4.00000000 6.00000000, 3.00000000 3.00000000)))
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "Representación de texto convencional (WKT)" en la página 529
- "Representación binaria convencional (WKB)" en la página 534
- "Representación de forma" en la página 536
- "Representación GML (Geography Markup Language)" en la página 536

ST_NumGeometries

ST_NumGeometries toma una colección de geometrías como parámetro de entrada y devuelve el número de geometrías de la colección.

Si la colección de geometrías proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

colección
Valor de tipo ST_GeomCollection o uno de sus subtipos que representa la colección de geometrías para la que se obtiene el número de geometrías.

Tipo de retorno:

INTEGER

Ejemplo:

La tabla SAMPLE_GEOMCOLL contiene dos colecciones de geometrías. Una es un multipolígono, y la otra es un multipunto. La función ST_NumGeometries determina cuántas geometrías individuales están contenidas dentro de cada colección de geometrías.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geomcoll (id INTEGER, geometry ST_GeomCollection)

INSERT INTO sample_geomcoll
VALUES( 1,
          ST_MultiPolygon ('multipolygon (( (3 3, 4 6, 5 3, 3 3),
                                (8 24, 9 25, 1 28, 8 24),
                                (13 33, 7 36, 1 40, 10 43, 13 33) ))', 1) )

INSERT INTO sample_geomcoll
  VALUES (2, ST_MultiPoint ('multipoint (1 2, 4 3, 5 6, 7 6, 8 8)', 1) )

SELECT id, ST_NumGeometries (geometry) NUM_GEOMS_IN_COLL
  FROM sample_geomcoll
```

Resultados:

```
ID          NUM_GEOMS_IN_COLL
----------- -----------------
          1                 3
          2                 5
```

Información relacionada:

- "ST_GeometryN" en la página 408

ST_NumInteriorRing

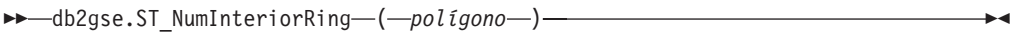
ST_NumInteriorRing toma un polígono como parámetro de entrada y devuelve el número de sus anillos interiores.

Si el polígono proporcionado es un valor nulo o está vacío, se devuelve un valor nulo.

Si el polígono no tiene anillos interiores, se devuelve un 0 (cero).

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

polígono
Valor de tipo ST_Polygon que representa el polígono para el que se devuelve el número de anillos interiores.

Tipo de retorno:

INTEGER

Ejemplo:

El ejemplo siguiente crea dos polígonos:

- Un polígono con dos anillos interiores

- Un polígono sin anillos interiores

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon('polygon
        ((40 120, 90 120, 90 150, 40 150, 40 120),
         (50 130, 60 130, 60 140, 50 140, 50 130),
         (70 130, 80 130, 80 140, 70 140, 70 130))' , 0) )

INSERT INTO sample_polys
  VALUES (2, ST_Polygon('polygon ((5 15, 50 15, 50 105, 5 15))' , 0) )
```

La función ST_NumInteriorRing se utiliza para obtener el número de anillos de las geometrías contenidas en la tabla:

```
SELECT id, ST_NumInteriorRing(geometry) NUM_RINGS
FROM sample_polys
```

Resultados:

```
ID         NUM_RINGS
---------- --------------
         1              2
         2              0
```

Información relacionada:

- "ST_InteriorRingN" en la página 415

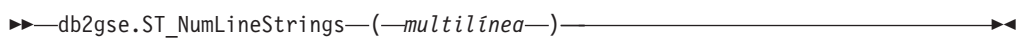
ST_NumLineStrings

ST_NumLineStrings toma una multilínea como parámetro de entrada y devuelve el número de cadenas lineales de la multilínea.

Si la multilínea proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

multilínea
Valor de tipo ST_MultiLineString que representa la multilínea para la que se devuelve el número de cadenas lineales.

Tipo de retorno:

INTEGER

Ejemplo:

La tabla SAMPLE_MLINES contiene multilíneas. La función ST_NumLineStrings determina cuántas geometrías individuales están contenidas dentro de cada multilínea.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mlines (id INTEGER, geometry ST_MultiLineString)

INSERT INTO sample_mlines
  VALUES (110, ST_MultiLineString ('multilinestring
       ( (33 2, 34 3, 35 6),
         (28 4, 29 5, 31 8, 43 12),
         (39 3, 37 4, 36 7))', 1) )
INSERT INTO sample_mlines
  VALUES (111, ST_MultiLineString ('multilinestring
       ( (3 2, 4 3, 5 6),
         (8 4, 9 5, 3 8, 4 12))', 1) )

SELECT id, ST_NumLineStrings (geometry) NUM_WITHIN
  FROM sample_mlines
```

Resultados:

```
ID          NUM_WITHIN
----------- ----------
        110          3
        111          2
```

Información relacionada:

- "ST_LineStringN" en la página 433

ST_NumPoints

ST_NumPoints toma una geometría como parámetro de entrada y devuelve el número de puntos que se utilizaron para definir esa geometría. Por ejemplo, si la geometría es un polígono y se utilizaron cinco puntos para definir ese polígono, el número devuelto es 5.

Si la geometría proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la que se obtiene el número de puntos.

Tipo de retorno:

INTEGER

Ejemplo:

La tabla contiene diversas geometrías. La función ST_NumPoints determina cuántos puntos hay dentro de cada geometría contenida en la tabla SAMPLE_GEOMETRIES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (spatial_type VARCHAR(18), geometry ST_Geometry)

INSERT INTO sample_geometries
  VALUES ('st_point',
           ST_Point (2, 3, 0) )

INSERT INTO sample_geometries
  VALUES ('st_linestring',
           ST_LineString ('linestring (2 5, 21 3, 23 10)', 0) )

INSERT INTO sample_geometries
  VALUES ('st_polygon',
           ST_Polygon ('polygon ((110 120, 110 140, 120 130, 110 120))', 0) )

SELECT spatial_type, ST_NumPoints (geometry) NUM_POINTS
  FROM sample_geometries
```

Resultados:

```
SPATIAL_TYPE    NUM_POINTS
--------------- ----------
st_point                 1
st_linestring            3
st_polygon               4
```

Información relacionada:

- "ST_PointN" en la página 483

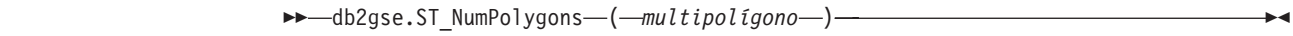
ST_NumPolygons

ST_NumPolygons toma un multipolígono como parámetro de entrada y devuelve el número de polígonos que contiene.

Si el multipolígono proporcionado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

multipolígono
Valor de tipo ST_MultiPolygon que representa el multipolígono para el que se obtiene el número de polígonos.

Tipo de retorno:

INTEGER

Ejemplo:

La tabla SAMPLE_MPOLYS contiene multipolígonos. La función ST_NumPolygons determina cuántas geometrías individuales están contenidas dentro de cada multipolígono.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER, geometry ST_MultiPolygon)

INSERT INTO sample_mpolys
VALUES( 1,
         ST_MultiPolygon ('multipolygon (( (3 3, 4 6, 5 3, 3 3),
                                (8 24, 9 25, 1 28, 8 24),
                                (13 33, 7 36, 1 40, 10 43, 13 33) ))', 1) )

INSERT INTO sample_polys
VALUES(2,
         ST_MultiPolygon ('multipolygon empty', 1) )

INSERT INTO sample_polys
  VALUES (3,
         ST_MultiPolygon ('multipolygon (( (3 3, 4 6, 5 3, 3 3),
                                (13 33, 7 36, 1 40, 10 43, 13 33) ))', 1) )

SELECT id, ST_NumPolygons (geometry) NUM_WITHIN
  FROM sample_mpolys
```

Resultados:

```
ID          NUM_WITHIN
----------- ----------
          1          3
          2          0
          3          2
```

Información relacionada:

- "ST_PolygonN" en la página 490

ST_Overlaps

ST_Overlaps toma dos geometrías como parámetros de entrada y devuelve un 1 si la intersección de las geometrías es una geometría de la misma dimensión pero que es diferente de las geometrías originales. En caso contrario, se devuelve un 0 (cero).

Si cualquiera de las dos geometrías es un valor nulo o está vacía, se devuelve un valor nulo.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Sintaxis:

Parámetros:

geometría1
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se comprueba si se solapa con *geometría2*.

geometría2
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se comprueba si se solapa con *geometría1*.

Tipo de retorno:

INTEGER

Ejemplos:

Estos ejemplos muestran el uso de ST_Overlaps. Se crean varias geometrías y luego se insertan en la tabla SAMPLE_GEOMETRIES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
  VALUES (1, ST_Point (10, 20, 1)),
         (2, ST_Point ('point (41 41)', 1) ),
         (10, ST_LineString ('linestring (1 10, 3 12, 10 10)', 1) ),
         (20, ST_LineString ('linestring (50 10, 50 12, 45 10)', 1) ),
         (30, ST_LineString ('linestring (50 12, 50 10, 60 8)', 1) ),
         (100, ST_Polygon ('polygon ((0 0, 0 40, 40 40, 40 0, 0 0))', 1) ),
         (110, ST_Polygon ('polygon ((30 10, 30 30, 50 30, 50 10, 30 10))', 1) ),
         (120, ST_Polygon ('polygon ((0 50, 0 60, 40 60, 40 60, 0 50))', 1) )
```

Ejemplo 1:

Este ejemplo determina los ID de puntos que se solapan.

```
SELECT sg1.id, sg2.id
  CASE ST_Overlaps (sg1.geometry, sg2.geometry)
    WHEN 0 THEN 'Points_do_not_overlap'
    WHEN 1 THEN 'Points_overlap'
    END
  AS OVERLAP
  FROM sample_geometries sg1, sample_geometries sg2
  WHERE sg1.id < 10 AND sg2.id < 10 AND sg1.id >= sg2.id
```

Resultados:

```
ID          ID         OVERLAP
----------- ---------- ------------------------
          1          1 Points_do_not_overlap
          2          1 Points_do_not_overlap
          2          2 Points_do_not_overlap
```

Ejemplo 2:

Este ejemplo determina los ID de líneas que se solapan.

```
SELECT sg1.id, sg2.id
  CASE ST_Overlaps (sg1.geometry, sg2.geometry)
    WHEN 0 THEN 'Lines_do_not_overlap'
    WHEN 1 THEN 'Lines_overlap'
    END
  AS OVERLAP
  FROM sample_geometries sg1, sample_geometries sg2
  WHERE sg1.id >= 10 AND sg1.id < 100
    AND sg2.id >= 10 AND sg2.id < 100
    AND sg1.id >= sg2.id
```

Resultados:

```
ID          ID         OVERLAP
----------- ---------- ------------------------
         10         10 Lines_do_not_overlap
         20         10 Lines_do_not_overlap
         30         10 Lines_do_not_overlap
         20         20 Lines_do_not_overlap
         30         20 Lines_overlap
         30         30 Lines_do_not_overlap
```

Ejemplo 3:

Este ejemplo determina los ID de polígonos que se solapan.

```
SELECT sg1.id, sg2.id
  CASE ST_Overlaps (sg1.geometry, sg2.geometry)
    WHEN 0 THEN 'Polygons_do_not_overlap'
    WHEN 1 THEN 'Polygons_overlap'
    END
  AS OVERLAP
  FROM sample_geometries sg1, sample_geometries sg2
  WHERE sg1.id >= 100 AND sg2.id >= 100 AND sg1.id >= sg2.id
```

Resultados:

```
ID          ID         OVERLAP
----------- ---------- ------------------------
        100        100 Polygons_do_not_overlap
        110        100 Polygons_overlap
        120        100 Polygons_do_not_overlap
        110        110 Polygons_do_not_overlap
        120        110 Polygons_do_not_overlap
        120        120 Polygons_do_not_overlap
```

Información relacionada:

• "Funciones que utilizan índices para optimizar consultas" en la página 128

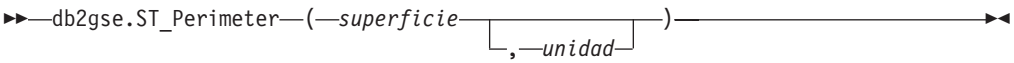
ST_Perimeter

ST_Perimeter toma como parámetros de entrada una superficie o multisuperficie y, opcionalmente, una unidad de medida, y devuelve el perímetro de la superficie o multisuperficie, que es la longitud de sus límites, expresado en la unidad de medida proporcionada o en la unidad por omisión.

Si la superficie o multisuperficie proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

superficie
Valor de tipo ST_Surface, ST_MultiSurface o uno de sus subtipos para el que se obtiene el perímetro.

unidad Valor de tipo VARCHAR(128) que identifica las unidades utilizadas para medir el perímetro. Las unidades de medida soportadas están listadas en la vista de catálogo DB2GSE.ST_UNITS_OF_MEASURE.

Si se omite el parámetro *unidad*, se siguen las normas siguientes para determinar la unidad utilizada para medir el perímetro:

- Si la *superficie* está en un sistema de coordenadas proyectadas o geocéntricas, se utiliza la unidad lineal correspondiente a este sistema de coordenadas por omisión.

- Si la *superficie* está en un sistema de coordenadas geográficas pero no en un sistemas de referencia espacial (SRS) geodésico, se utiliza la unidad angular correspondiente a este sistema de coordenadas por omisión.
- Si la *superficie* está en un SRS geodésico, la unidad de medida por omisión será el metro.

Restricciones para las conversiones de unidades: Se devuelve un error (SQLSTATE 38SU4) si se cumple alguna de las siguientes condiciones:

- La geometría está en un sistema de coordenadas no especificado y se especifica el parámetro *unidad*.
- La geometría está en un sistema de coordenadas proyectadas y se especifica una unidad angular.
- La geometría está en un sistema de coordenadas geográficas pero no en un SRS geodésico y se ha especificado una unidad lineal.
- La geometría está en un sistema de coordenadas geográficas, está en un SRS geodésico y se especifica una unidad angular.

Tipo de retorno:

DOUBLE

Ejemplos:

Estos ejemplos muestran el uso de la función ST_Perimeter. Se invoca a db2se para crear un sistema de referencia espacial cuyo ID es 4000, y se crea un polígono en ese sistema de referencia espacial.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

db2se create_srs se_bank -srsId 4000 -srsName new_york1983
  -xOffset 0 -yOffset 0 -xScale 1 -yScale 1
  -coordsysName NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet
```

La tabla SAMPLE_POLYS se crea para contener una geometría cuyo perímetro es 18.

```
CREATE TABLE sample_polys (id SMALLINT, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon ('polygon ((0 0, 0 4, 5 4, 5 0, 0 0))', 4000))
```

Ejemplo 1:

Este ejemplo obtiene el ID y el perímetro del polígono.

```
SELECT id, ST_Perimeter (geometry) AS PERIMETER
  FROM sample_polys
```

Resultados:

```
ID          PERIMETER
----------- ------------------------
          1 +1.80000000000000E+001
```

Ejemplo 2:

Este ejemplo obtiene el ID y el perímetro del polígono, con el perímetro expresado en metros.

```
SELECT id, ST_Perimeter (geometry, 'METER') AS PERIMETER_METER
  FROM sample_polys
```

Resultados:

```
ID          PERIMETER_METER
----------- ----------------------
          1 +5.48641097282195E+000
```

ST_PerpPoints

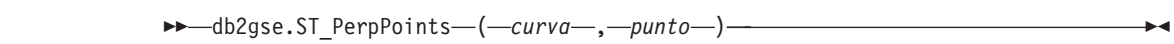
ST_PerpPoints toma como parámetros de entrada una curva o multicurva y un punto y devuelve la proyección perpendicular de ese punto en la curva o multicurva. Se obtiene el punto con la distancia más pequeña entre el punto proporcionado y el punto perpendicular. Si dos o más puntos proyectados perpendicularmente son equidistantes respecto al punto proporcionado, se devuelven todos ellos. Si no puede crearse ningún punto perpendicular, se devuelve un punto vacío.

Si la curva o multicurva proporcionada tiene coordenadas Z o M, la coordenada Z o M de los puntos resultantes se calcula por interpolación sobre la curva o multicurva proporcionada.

Si la curva o punto proporcionados están vacíos, se devuelve un punto vacío. Si la curva o punto proporcionados es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

curva Valor de tipo ST_Curve, ST_MultiCurve o uno de sus subtipos que representa la curva o multicurva en la que se obtiene la proyección perpendicular del *punto*.

punto Un valor de tipo ST_Point que representa el punto que se proyecta perpendicularmente sobre *curva*.

Tipo de retorno:

db2gse.ST_MultiPoint

Ejemplos:

Estos ejemplos muestran el uso de la función ST_PerpPoints para encontrar puntos que son perpendiculares a la cadena lineal contenida en la tabla siguiente. La función ST_LineString se utiliza en la sentencia INSERT para crear la cadena lineal.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)

INSERT INTO sample_lines (id, line)
  VALUES (1, ST_LineString('linestring (0 10, 0 0, 10 0, 10 10)' , 0) )
```

Ejemplo 1:

Este ejemplo obtiene la proyección perpendicular sobre la cadena lineal de un punto cuyas coordenadas son (5, 0). La función ST_AsText se utiliza para convertir el valor devuelto (un multipunto) en su correspondiente representación de texto convencional.

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 0) ) )
  AS VARCHAR(50) ) PERP
  FROM sample_lines
```

Resultados:

```
PERP
----------------------------------------
MULTIPOINT ( 5.00000000 0.00000000)
```

Ejemplo 2:

Este ejemplo obtiene la proyección perpendicular sobre la cadena lineal de un punto cuyas coordenadas son (5, 5). En este caso, existen tres puntos en la cadena lineal que son equidistantes de la ubicación proporcionada. Por tanto, se obtiene un multipunto formado por los tres puntos.

```
SELECT  CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 5) ) )
  AS VARCHAR160) ) PERP
  FROM sample_lines
```

Resultados:

```
PERP
--------------------------------------------------------------------------------
MULTIPOINT ( 0.00000000 5.00000000, 5.00000000 0.00000000, 10.00000000 5.00000000)
```

Ejemplo 3:

Este ejemplo obtiene la proyección perpendicular sobre la cadena lineal de un punto cuyas coordenadas son (5, 10). En este caso, se pueden encontrar tres puntos perpendiculares diferentes. Pero la función ST_PerpPoints sólo devuelve los puntos que están más cerca del punto proporcionado. Por tanto, se obtiene un multipunto formado solamente por los dos puntos más cercanos. El tercer punto no se incluye.

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 10) ) )
  AS VARCHAR(80) ) PERP
  FROM sample_lines
```

Resultados:

```
PERP
------------------------------------------------------------
MULTIPOINT ( 0.00000000 10.00000000, 10.00000000 10.00000000 )
```

Ejemplo 4:

Este ejemplo obtiene la proyección perpendicular sobre la cadena lineal de un punto cuyas coordenadas son (5, 15).

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point('point(5 15)', 0 ) ) )
  AS VARCHAR(80) ) PERP
  FROM sample_lines
```

Resultados:

```
PERP
-----------------------------------
MULTIPOINT ( 5.00000000 0.00000000 )
```

Ejemplo 5:

En este ejemplo, el punto especificado cuyas coordenadas son (15 15) carece de proyección perpendicular sobre la cadena lineal. Por tanto, se devuelve una geometría vacía.

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(15, 15) ) )
  AS VARCHAR(80) ) PERP
  FROM sample_lines
```

Resultados:

```
PERP
--------------------------------------------
MULTIPOINT EMPTY
```

ST_Point

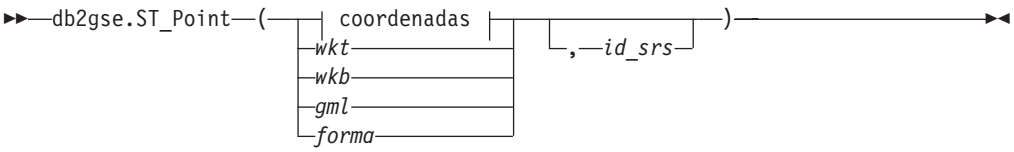
ST_Point obtiene un punto a partir de uno de estos conjuntos de datos de entrada:

- Coordenadas X e Y solamente
- Coordenadas X, Y y Z
- Coordenadas X, Y, Z y M
- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma
- Una representación en formato GML (Geography Markup Language)

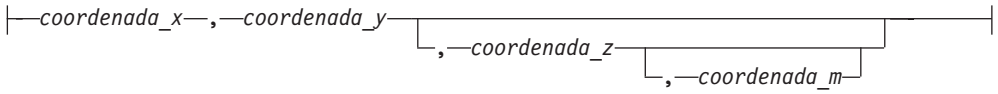
Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenido el punto resultante.

Si el punto se crea a partir de coordenadas, y si la coordenada X o Y es nula, se emite una condición de excepción (SQLSTATE 38SUP). Si la coordenada Z o M es nula, el punto resultante no tendrá coordenada Z o M, respectivamente. Si el punto se crea a partir de su representación de texto convencional, su representación binaria convencional, su representación de forma o su representación GML, y esta representación es nula, se devuelve un valor nulo.

Sintaxis:



coordenadas:



Parámetros:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional del punto resultante.

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional del punto resultante.

gml Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), del punto resultante.

forma Valor de tipo BLOB(2G) que contiene la representación de forma del punto resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al punto resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

coordenada_x
Valor de tipo DOUBLE que especifica la coordenada X del punto resultante.

coordenada_y
Valor de tipo DOUBLE que especifica la coordenada Y del punto resultante.

coordenada_z
Valor de tipo DOUBLE que especifica la coordenada Z del punto resultante.

Si se omite el parámetro *coordenada_z*, el punto resultante no tendrá coordenada Z. Para ese punto, el resultado de ST_Is3D será un 0 (cero).

coordenada_m
Valor de tipo DOUBLE que especifica la coordenada M del punto resultante.

Si se omite el parámetro *coordenada_m*, el punto resultante no tendrá una medida. Para ese punto, el resultado de ST_IsMeasured será un 0 (cero).

Tipo de retorno:

db2gse.ST_Point

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Ejemplo 1:

Este ejemplo muestra el uso de ST_Point para crear e insertar puntos. El primer punto se crea utilizando un conjunto de coordenadas X e Y. El segundo punto se crea utilizando su representación de texto convencional. Ambos puntos son geometrías que están contenidas en el sistema de referencia espacial 1.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points
  VALUES (1100, ST_Point (10, 20, 1) )

INSERT INTO sample_points
  VALUES (1101, ST_Point ('point (30 40)', 1) )
```

La sentencia SELECT siguiente obtiene los puntos que se registraron en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(90)) POINTS
 FROM sample_points
```

Resultados:

```
ID         POINTS
---------- ----------------------------------
      1110 POINT ( 10.00000000 20.00000000)
      1101 POINT ( 30.00000000 40.00000000)
```

Ejemplo 2:

Este ejemplo inserta un registro en la tabla SAMPLE_POINTS cuyo ID es 1103 y un valor de punto que tiene una coordenada X igual a 120, una coordenada Y igual a 358, una coordenada M igual a 34, y carece de coordenada Z.

```
INSERT INTO SAMPLE_POINTS(ID, GEOMETRY)
 VALUES(1103, db2gse.ST_Point(120, 358, CAST(NULL AS DOUBLE), 34, 1))
 SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(90) ) POINTS
 FROM sample_points
```

Resultados:

```
ID         POINTS
 ---------- -----------------------------------------------
       1103 POINT M ( 120.0000000 358.0000000 34.00000000)
```

Ejemplo 3:

Este ejemplo inserta una fila en la tabla SAMPLE_POINTS cuyo ID es 1104 y un valor de punto que tiene una coordenada X igual a 1003, una coordenada Y igual a 9876, una coordenada Z igual a 20, en el sistema de referencia espacial 0, y utilizando la representación GML.

```
INSERT INTO SAMPLE_POINTS(ID, GEOMETRY)
 VALUES(1104, db2gse.ST_Point('<gml:Point><gml:coord>
    <gml:x>1003</gml:X><gml:Y>9876</gml:Y><gml:Z>20</gml:Z>
    </gml:coord></gml:Point>', 1))
 SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(90) ) POINTS
 FROM sample_points
```

Resultados:

```
ID         POINTS
 ---------- ------------------------------------------------
       1104 POINT Z ( 1003.000000 9876.000000 20.00000000)
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_Is3d" en la página 419
- "ST_IsMeasured" en la página 423
- "Representación de texto convencional (WKT)" en la página 529
- "Representación binaria convencional (WKB)" en la página 534
- "Representación de forma" en la página 536
- "Representación GML (Geography Markup Language)" en la página 536

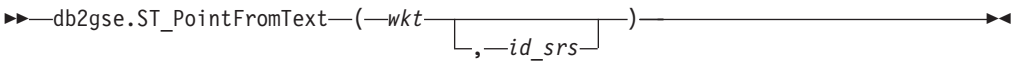
ST_PointFromText

ST_PointFromText toma como parámetros de entrada una representación de texto convencional de un punto y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el punto correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST_Point. Es recomendable debido a su flexibilidad: ST_Point acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

Sintaxis:



Parámetros:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional del punto resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al punto resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_Point

Ejemplo:

Este ejemplo muestra cómo utilizar ST_PointFromText para crear e insertar un punto a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es un punto representado en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son: (10, 20).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points
  VALUES (1110, ST_PointFromText ('point (30 40)', 1) )
```

La sentencia SELECT siguiente devuelve el polígono que se registró en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(35) ) POINTS
  FROM sample_points
  WHERE id = 1110
```

Resultados:

```
ID         POINTS
---------- -------------------------------------
      1110 POINTS ( 30.00000000 40.00000000)
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_Point" en la página 478
- "ST_PointFromWKB" en la página 482

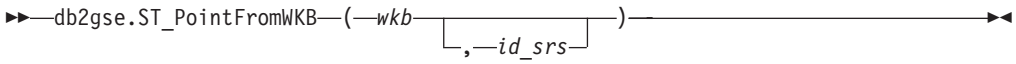
ST_PointFromWKB

ST_PointFromWKB toma como parámetros de entrada una representación binaria convencional de un punto y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el punto correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST_Point. Es recomendable debido a su flexibilidad: ST_Point acepta otras modalidades de datos de entrada, además de la representación binaria convencional.

Sintaxis:



Parámetros:

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional del punto resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al punto resultante.

Si se omite el parámetro *id_srs*, se utiliza implícitamente el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_Point

Ejemplo:

Este ejemplo muestra cómo utilizar ST_PointFromWKB para crear un punto a partir de su representación binaria convencional. Las geometrías son puntos en el sistema de referencia espacial 1. En este ejemplo, los puntos se guardan en la columna GEOMETRY de la tabla SAMPLE_POLYS, y luego la columna WKB se actualiza con su representación binaria convencional (utilizando la función ST_AsBinary). Finalmente, se utiliza la función ST_PointFromWKB para obtener los puntos a partir de la columna WKB.

La tabla SAMPLE_POINTS tiene una columna GEOMETRY, donde se guardan los puntos, y una columna WKB, donde se guardan las representaciones binarias convencionales de los puntos.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point, wkb BLOB(32K))

INSERT INTO sample_points
  VALUES (10, ST_Point ('point (44 14)', 1) ),
  VALUES (11, ST_Point ('point (24 13)', 1))

UPDATE sample_points AS temporary_correlated
  SET wkb = ST_AsBinary( geometry )
  WHERE id = temporary_correlated.id
```

En la sentencia SELECT siguiente, se utiliza la función ST_PointFromWKB para obtener los puntos a partir de la columna WKB.

```
SELECT id, CAST( ST_AsText( ST_PolyFromWKB (wkb) ) AS VARCHAR(35) ) POINTS
  FROM sample_points
```

Resultados:

```
ID         POINTS
---------- -----------------------------------
        10 POINT ( 44.00000000 14.00000000)
        11 POINT ( 24.00000000 13.00000000)
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_Point" en la página 478
- "ST_PointFromText" en la página 481

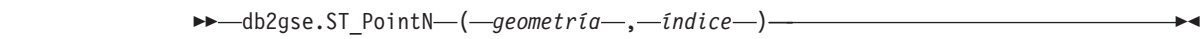
ST_PointN

ST_PointN toma una cadena lineal o un multipunto, y un índice como parámetros de entrada y devuelve el punto identificado por el índice y perteneciente a la cadena lineal o multipunto. El punto resultante se representa según el sistema de referencia espacial de la cadena lineal o multipunto proporcionados.

Si la cadena lineal o multipunto proporcionados es un valor nulo o está vacío, se devuelve un valor nulo. Si el índice es menor que 1 o mayor que el número de puntos de la cadena lineal o multipunto, se devuelve un valor nulo y se emite una condición de aviso (SQLSTATE 01HS2).

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

geometría
: Un valor de tipo ST_LineString o ST_MultiPoint que representa la geometría a partir de la que se devuelve el punto que está identificado mediante *índice*.

índice Valor de tipo INTEGER que identifica el punto *n* que se debe obtener a partir de *geometría*.

Tipo de retorno:

db2gse.ST_Point

Ejemplo:

El ejemplo siguiente muestra el uso de ST_PointN.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)

INSERT INTO sample_lines
  VALUES (1, ST_LineString ('linestring (10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0) )

SELECT id, CAST ( ST_AsText (ST_PointN (line, 2) ) AS VARCHAR(60) ) SECOND_INDEX
  FROM sample_lines
```

Resultados:

```
ID        SECOND_INDEX
--------- ------------------------------
        1 POINT (5.00000000 5.00000000)
```

Información relacionada:

- "ST_Endpoint" en la página 389
- "ST_NumPoints" en la página 470
- "ST_StartPoint" en la página 496

ST_PointOnSurface

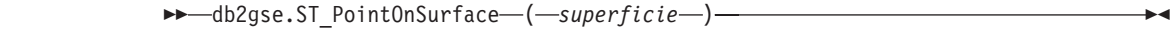
ST_PointOnSurface toma como parámetro de entrada una superficie o multisuperficie y devuelve un punto que con toda seguridad estará en el interior de la superficie o multisuperficie. Este punto es el paracentro de la superficie.

El punto resultante se representa según el sistema de referencia espacial de la superficie o multisuperficie proporcionada.

Si la superficie o multisuperficie proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

superficie
Valor de tipo ST_Surface, ST_MultiSurface o uno de sus subtipos que representa la geometría para la que se obtiene un punto situado en la superficie.

Tipo de retorno:

db2gse.ST_Point

Ejemplo:

El ejemplo siguiente crea dos polígonos y luego utiliza la función ST_PointOnSurface. Uno de los polígonos tiene un hueco en su centro. Los puntos obtenidos están situados en la superficie de los polígonos. No se encuentran necesariamente en el centro exacto de los polígonos.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
VALUES( 1,
          ST_Polygon ('polygon ( (40 120, 90 120, 90 150, 40 150, 40 120) ,
                           (50 130, 80 130, 80 140, 50 140, 50 130) )' ,0) )
INSERT INTO sample_polys
VALUES(2,
          ST_Polygon ('polygon ( (10 10, 50 10, 10 30, 10 10) )', 0) )

SELECT id, CAST (ST_AsText (ST_PointOnSurface (geometry) ) AS VARCHAR(80) )
  POINT_ON_SURFACE
  FROM sample_polys
```

Resultados:

```
ID          POINT_ON_SURFACE
----------- ------------------------------------
          1 POINT ( 65.00000000 125.00000000)
          2 POINT ( 30.00000000 15.00000000)
```

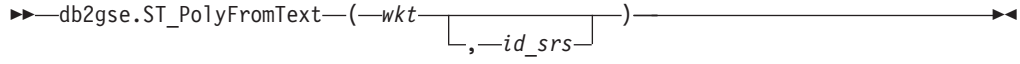
ST_PolyFromText

ST_PolyFromText toma como parámetros de entrada una representación de texto convencional de un polígono y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el polígono correspondiente.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST_Polygon. Es recomendable debido a su flexibilidad: ST_Polygon acepta otras modalidades de datos de entrada, además de la representación de texto convencional.

Sintaxis:



Parámetros:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional del polígono resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al polígono resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_Polygon

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST_PolyFromText para crear e insertar un polígono a partir de su representación de texto convencional. El registro que se inserta tiene un ID = 1110, y la geometría es un polígono representado en el sistema de referencia espacial 1 con una representación de texto convencional. Las coordenadas X e Y de esta geometría son: (50, 20) (50, 40) (70, 30).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1110, ST_PolyFromText ('polygon ((50 20, 50 40, 70 30, 50 20))', 1) )
```

La sentencia SELECT siguiente devuelve el polígono que se registró en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(120) ) POLYGON
  FROM sample_polys
  WHERE id = 1110
```

Resultados:

```
ID         POLYGON
---------- ------------------------------------------------------------------
      1110 POLYGON (( 50.00000000 20.00000000, 70.00000000 30.00000000,
                      50.00000000 40.00000000, 50.00000000 20.00000000))
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_PolyFromWKB" en la página 486
- "ST_Polygon" en la página 488

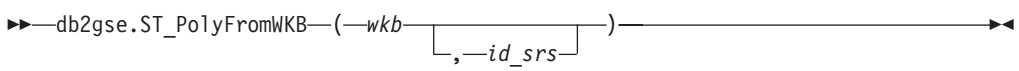
ST_PolyFromWKB

ST_PolyFromWKB toma como parámetros de entrada una representación binaria convencional de un polígono y, opcionalmente, un identificador de sistemas de referencia espacial y devuelve el polígono correspondiente.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

La función recomendada para obtener el mismo resultado es ST_Polygon. Es recomendable debido a su flexibilidad: ST_Polygon acepta otras modalidades de datos de entrada, además de la representación binaria convencional.

Sintaxis:



Parámetros:

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional del polígono resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al polígono resultante.

Si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_Polygon

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra cómo utilizar ST_PolyFromWKB para crear un polígono a partir de su representación binaria convencional. La geometría es un polígono en el sistema de referencia espacial 1. En este ejemplo, el polígono se guarda con el ID = 1115 en la columna GEOMETRY de la tabla SAMPLE_POLYS, y luego la columna WKB se actualiza con su representación binaria convencional (utilizando la función ST_AsBinary). Finalmente, se utiliza la función ST_PolyFromWKB para obtener el multipolígono a partir de la columna WKB. Las coordenadas X e Y de esta geometría son: (50, 20) (50, 40) (70, 30).

La tabla SAMPLE_POLYS tiene una columna GEOMETRY, donde se guarda el polígono, y una columna WKB, donde se guarda la representación binaria convencional de polígono.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon,
  wkb BLOB(32K))

INSERT INTO sample_polys
  VALUES (10, ST_Polygon ('polygon ((50 20, 50 40, 70 30, 50 20))', 1) )

UPDATE sample_polys AS temporary_correlated
  SET wkb = ST_AsBinary( geometry )
  WHERE id = temporary_correlated.id
```

En la sentencia SELECT siguiente, se utiliza la función ST_PolyFromWKB para obtener el polígono a partir de la columna WKB.

```
SELECT id, CAST( ST_AsText( ST_PolyFromWKB (wkb) )
  AS VARCHAR(120) ) POLYGON
  FROM sample_polys
  WHERE id = 1115
```

Resultados:

```
ID         POLYGON
---------- ------------------------------------------------------------
      1115 POLYGON (( 50.00000000 20.00000000, 70.00000000
             30.00000000,50.00000000 40.00000000, 50.00000000
             20.00000000))
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_PolyFromText" en la página 485
- "ST_Polygon" en la página 488

ST_Polygon

ST_Polygon crea un polígono a partir de uno de estos datos de entrada:

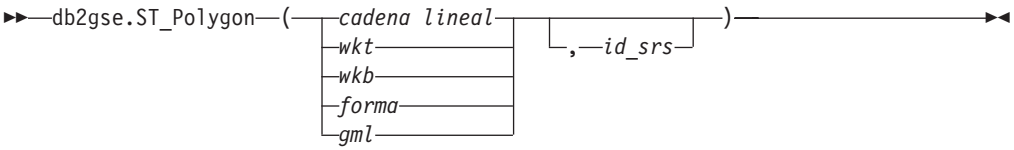
- Una cadena lineal cerrada que define el anillo exterior del polígono resultante
- Una representación de texto convencional
- Una representación binaria convencional
- Una representación de forma
- Una representación en formato GML (Geography Markup Language)

Opcionalmente, se puede especificar un identificador para identificar el sistema de referencia espacial donde está contenido el polígono resultante.

Si el polígono se crea a partir de una cadena lineal y ésta es un valor nulo, se devuelve un valor nulo. Si la cadena lineal proporcionada está vacía, se obtiene un polígono vacío. Si el polígono se crea a partir de su representación de texto convencional, su representación binaria convencional, su representación de forma o su representación GML, y la representación es nula, se devuelve un valor nulo.

También se puede invocar a esta función como método para los casos siguientes solamente: ST_Polygon(*cadena lineal*) y ST_Polygon(*cadena lineal*, *id_srs*).

Sintaxis:



Parámetros:

cadena lineal
: Valor de tipo ST_LineString que representa la cadena lineal que define el anillo exterior correspondiente al perímetro externo. Si *cadena lineal* no es cerrada y es simple, se emite una condición de excepción (SQLSTATE 38SSL).

wkt
: Valor de tipo CLOB(2G) que contiene la representación de texto convencional del polígono resultante.

wkb
: Valor de tipo BLOB(2G) que contiene la representación binaria convencional del polígono resultante.

forma Valor de tipo BLOB(2G) que contiene la representación de forma del polígono resultante.

gml Valor de tipo CLOB(2G) que contiene la representación, en formato GML (Geography Markup Language), del polígono resultante.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente al polígono resultante.

Si el polígono se crea a partir de un parámetro *cadena lineal* proporcionado y se omite el parámetro *id_srs*, se utiliza implícitamente el sistema de referencia espacial de *cadena lineal*. En otro caso, si se omite el parámetro *id_srs*, se utiliza el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_Polygon

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de ST_Polygon para crear e insertar polígonos. Se crean e insertan tres polígonos. Todos ellos son geometrías que están contenidas en el sistema de referencia espacial 1.

- El primer polígono se crea a partir de un anillo (una cadena lineal cerrada y simple). Las coordenadas X e Y de este polígono son: (10, 20) (10, 40) (20, 30).
- El segundo polígono se crea utilizando su representación de texto convencional. Las coordenadas X e Y de este polígono son: (110, 120) (110, 140) (120, 130).
- El tercer polígono es un polígono con un hueco en su centro. Un polígono con un hueco en el centro consta de un polígono interior y un polígono exterior. Este polígono con un hueco en el centro se crea utilizando su representación de texto convencional. Las coordenadas X e Y del polígono exterior son: (110, 120) (110, 140) (130, 140) (130, 120) (110, 120). Las coordenadas X e Y del polígono interior son: (115, 125) (115, 135) (125, 135) (125, 135) (115, 125).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1100,
          ST_Polygon (ST_LineString ('linestring
                     (10 20, 10 40, 20 30, 10 20)',1), 1))

INSERT INTO sample_polys
  VALUES (1101,
          ST_Polygon ('polygon
                    ((110 120, 110 140, 120 130, 110 120))', 1))

INSERT INTO sample_polys
  VALUES (1102,
          ST_Polygon ('polygon
                    ((110 120, 110 140, 130 140, 130 120, 110 120),
                    (115 125, 115 135, 125 135, 125 135, 115 125))', 1))
```

La sentencia SELECT siguiente obtiene los polígonos que se registraron en la tabla:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(120) ) POLYGONS
  FROM sample_polys
```

Resultados:

```
ID          POLYGONS
------- ------------------------------------------------------------
   1110 POLYGON (( 10.00000000 20.00000000, 20.00000000 30.00000000
                   10.00000000 40.00000000, 10.00000000 20.00000000))

   1101 POLYGON (( 110.00000000 120.00000000, 120.00000000 130.00000000
                   110.00000000 140.00000000, 110.00000000 120.00000000))

   1102 POLYGON (( 110.00000000 120.00000000, 130.00000000 120.00000000
                   130.00000000 140.00000000, 110.00000000 140.00000000
                   110.00000000 120.00000000),
                 ( 115.00000000 125.00000000, 115.00000000 135.00000000
                   125.00000000 135.00000000, 125.00000000 135.00000000
                   115.00000000 125.00000000))
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "Representación de texto convencional (WKT)" en la página 529
- "Representación binaria convencional (WKB)" en la página 534
- "Representación de forma" en la página 536
- "Representación GML (Geography Markup Language)" en la página 536

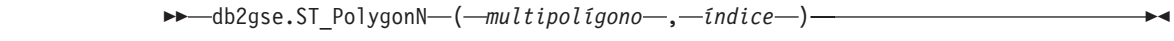
ST_PolygonN

ST_PolygonN toma un multipolígono y un índice como parámetros de entrada y devuelve el polígono identificado por el índice. El polígono resultante se representa según el sistema de referencia espacial del multipolígono proporcionado.

Si el multipolígono proporcionado es nulo o está vacío, o si el índice es menor que 1 o mayor que el número de polígonos, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

multipolígono
: Valor de tipo ST_MultiPolygon que representa el multipolígono a partir del cual se obtiene el polígono identificado por *índice*.

índice
: Valor de tipo INTEGER que identifica el polígono que ocupa la posición *n* y que se debe obtener a partir de *multipolígono*.

Tipo de retorno:

db2gse.ST_Polygon

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de ST_PolygonN.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER, geometry ST_MultiPolygon)

INSERT INTO sample_mpolys
  VALUES (1, ST_Polygon ('multipolygon (((3 3, 4 6, 5 3, 3 3),
                                         (8 24, 9 25, 1 28, 8 24)
                                         (13 33, 7 36, 1 40, 10 43,
                                          13 33)))', 1))

SELECT id, CAST ( ST_AsText (ST_PolygonN (geometry, 2) )
    AS VARCHAR(120) ) SECOND_INDEX
FROM sample_mpolys
```

Resultados:

```
ID        SECOND_INDEX
--------- ------------------------------------------------------------
        1 POLYGON (( 8.00000000 24.00000000, 9.00000000 25.00000000,
                     1.00000000 28.00000000, 8.00000000 24.00000000))
```

Información relacionada:

- "ST_NumPolygons" en la página 471

ST_Relate

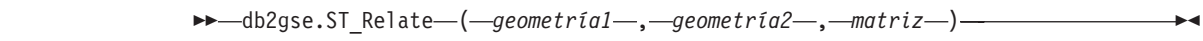
ST_Relate toma dos geometrías y una matriz DE-9IM como parámetros de entrada y devuelve 1 si dichas geometrías cumplen las condiciones especificadas por la matriz. En caso contrario, se devuelve un 0 (cero).

Si cualquiera de las geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

geometría1
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se compara con *geometría2*.

geometría2
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se compara con *geometría1*.

matriz Valor de tipo CHAR(9) que representa la matriz DE-9IM (Dimensionally Extended 9 Intersection Model) que se utilizará para la verificación de *geometría1* y *geometría2*.

Tipo de retorno:

INTEGER

Ejemplo:

El ejemplo siguiente crea dos polígonos separados. Luego, se utiliza la función ST_Relate para determinar diversas relaciones entre los dos polígonos. Por ejemplo, se determina si los dos polígonos se solapan.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
VALUES( 1,
         ST_Polygon('polygon ( (40 120, 90 120, 90 150, 40 150, 40 120) )', 0))
INSERT INTO sample_polys
VALUES(2,
         ST_Polygon('polygon ( (30 110, 50 110, 50 130, 30 130, 30 110) )', 0))

SELECT ST_Relate(a.geometry, b.geometry, 'T*T***T**')  "Overlaps  ",
       ST_Relate(a.geometry, b.geometry, 'T*T***FF*')  "Contains  ",
       ST_Relate(a.geometry, b.geometry, 'T*F**F***')  "Within    "
       ST_Relate(a.geometry, b.geometry, 'T********')  "Intersects",
       ST_Relate(a.geometry, b.geometry, 'T*F**FFF2')  "Equals    "
  FROM sample_polys a, sample_polys b
  WHERE a.id = 1 AND b.id = 2
```

Resultados:

```
Overlaps    Contains    Within      Intersects  Equals
----------- ----------- ----------- ----------- -----------
          1           0           0           1           0
```

Información relacionada:

- "Funciones que comparan elementos geográficos" en la página 316

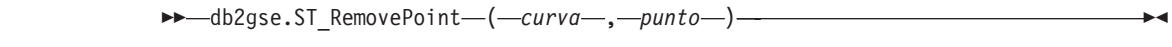
ST_RemovePoint

ST_RemovePoint toma una curva y un punto como parámetros de entrada y devuelve dicha curva después de eliminar de ella todos los puntos que son iguales al punto especificado. Si la curva proporcionada tiene coordenadas Z o M, el punto también debe tener coordenadas Z o M. La geometría resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

Si la curva proporcionada está vacía, se devuelve una curva vacía. Si la curva proporcionada es un valor nulo, o si el punto proporcionado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

curva Valor de tipo ST_Curve o uno de sus subtipos que representa la curva de la cual se elimina *punto*.

punto Valor de tipo ST_Point que identifica los puntos que se eliminan de *curva*.

Tipo de retorno:

db2gse.ST_Curve

Ejemplos:

El ejemplo siguiente añade dos cadenas lineales a la tabla SAMPLE_LINES. Estas cadenas lineales se utilizan en los ejemplos mostrados más abajo.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)

INSERT INTO sample_lines
  VALUES (1,ST_LineString('linestring
         (10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0))

INSERT INTO sample_lines
  VALUES (2, ST_LineString('linestring z
         (0 0 4, 5 5 5, 10 10 6, 5 5 7, 0 0 8)', 0))
```

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Ejemplo 1:

El ejemplo siguiente elimina el punto (5, 5) de la cadena lineal cuyo ID es 1. Este punto aparece dos veces en la cadena lineal. Por tanto, se eliminan ambas apariciones.

```
SELECT CAST(ST_AsText (ST_RemovePoint (line, ST_Point(5, 5) ) )
  AS VARCHAR(120) ) RESULT
  FROM sample_lines
  WHERE id = 1
```

Resultados:

```
RESULT
------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 0.00000000 0.00000000,
             10.00000000 0.00000000, 0.00000000 10.00000000)
```

Ejemplo 2:

El ejemplo siguiente elimina el punto (5, 5, 5) de la cadena lineal cuyo ID es 2. Este punto aparece una sola vez, por lo que sólo se elimina esa aparición.

```
SELECT CAST (ST_AsText (ST_RemovePoint (line, ST_Point(5.0, 5.0, 5.0)))
  AS VARCHAR(160) ) RESULT
  FROM sample_lines
  WHERE id=2
```

Resultados:

```
RESULT
------------------------------------------------------------------------
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 10.00000000 10.00000000
               6.00000000,  5.00000000 5.00000000 7.00000000, 0.00000000
               0.00000000 8.00000000)
```

ST_SrsId, ST_SRID

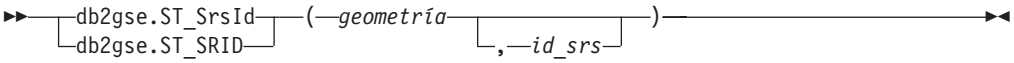
ST_SrsId (o ST_SRID) toma como parámetros de entrada una geometría y, opcionalmente, un identificador de sistemas de referencia espacial. Los datos devueltos dependen de los parámetros de entrada que se hayan especificado:

- Si se especifica el identificador de sistemas de referencia espacial, la función devuelve la geometría con su identificador cambiado al identificador especificado. No se realiza ninguna transformación de la geometría.
- Si no se proporciona como parámetro de entrada ningún identificador de sistemas de referencia espacial, se devuelve el identificador actual de la geometría especificada.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

Estas funciones se pueden invocar también como métodos.

Sintaxis:



Parámetros:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la que se define u obtiene el identificador de sistemas de referencia espacial.

id_srs Valor de tipo INTEGER que identifica el sistema de referencia espacial que se utilizará para la geometría resultante.

Atención: Si se especifica este parámetro, la geometría no se transforma, sino que se devuelve con su sistema de referencia espacial cambiado al sistema de referencia espacial especificado. Como resultado del cambio al nuevo sistema de referencia espacial, los datos pueden estar corrompidos. Para las transformaciones, utilice ST_Transform.

Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipos de retorno:

- INTEGER, si no se especifica un id_srs
- db2gse.ST_Geometry, si se especifica un id_srs

Ejemplo:

Este ejemplo crea dos puntos en dos sistemas de referencia espacial diferentes. El ID del sistema de referencia espacial correspondiente a cada punto se puede determinar utilizando la función ST_SrsId.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points
  VALUES (1, ST_Point( 'point (80 180)', 0 ) )
```

```
INSERT INTO sample_points
  VALUES (2, ST_Point( 'point (-74.21450127 + 42.03415094)', 1 ) )

SELECT id, ST_SRSId (geometry) SRSID
  FROM sample_points
```

Resultados:

```
ID          SRSID
----------- -----------
          1           0
          2           1
```

Información relacionada:

- "ST_Transform" en la página 508

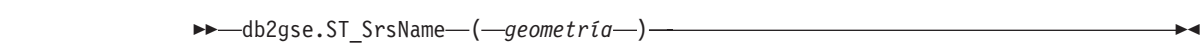
ST_SrsName

ST_SrsName toma una geometría como parámetro de entrada y devuelve el nombre del sistema de referencia espacial en el que está representada dicha geometría.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la que se devuelve el nombre del sistema de referencia espacial.

Tipo de retorno:

VARCHAR(128)

Ejemplo:

Este ejemplo crea dos puntos en sistemas de referencia espacial diferentes. Se utiliza la función ST_SrsName para determinar el nombre del sistema de referencia espacial correspondiente a cada punto.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry, ST_Point)

INSERT INTO sample_points
  VALUES (1, ST_Point ('point (80 180)', 0) )

INSERT INTO sample_points
  VALUES (2, ST_Point ('point (-74.21450127 + 42.03415094)', 1) )

SELECT id, ST_SrsName (geometry) SRSNAME
  FROM sample_points
```

Resultados:

```
ID          SRSNAME
----------- ------------------------------
          1 DEFAULT_SRS
          2 NAD83_SRS_1
```

Información relacionada:

- "ST_SrsId, ST_SRID" en la página 494

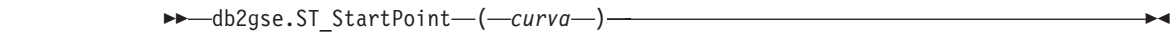
ST_StartPoint

ST_StartPoint toma una curva como parámetro de entrada y devuelve el primer punto de la curva. El punto resultante se representa según el sistema de referencia espacial de la curva proporcionada. Este resultado es equivalente al obtenido cuando se invoca a la función ST_PointN(*curva*, 1)

Si la curva proporcionada es un valor nulo o está vacía, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

curva Valor de tipo ST_Curve o uno de sus subtipos que representa la geometría a partir de la cual se obtiene el primer punto.

Tipo de retorno:

db2gse.ST_Point

Ejemplo:

El ejemplo siguiente añade dos cadenas lineales a la tabla SAMPLE_LINES. La primera cadena lineal tiene coordenadas X e Y. La segunda cadena lineal tiene coordenadas X, Y y Z. Se utiliza la función ST_StartPoint para obtener el primer punto de cada cadena lineal.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)

INSERT INTO sample_lines
  VALUES (1, ST_LineString ('linestring
         (10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0))

INSERT INTO sample_lines
  VALUES (1, ST_LineString ('linestring z
         (0 0 4, 5 5 5, 10 10 6, 5 5 7, 0 0 8)', 0))

SELECT id, CAST( ST_AsText( ST_StartPoint( line ) ) AS VARCHAR(80))
  START_POINT
  FROM sample_lines
```

Resultados:

```
ID          START_POINT
----------- ----------------------------------------
          1 POINT ( 10.00000000 10.00000000)
          2 POINT Z ( 0.00000000 0.00000000 4.00000000)
```

Información relacionada:

- "ST_Endpoint" en la página 389
- "ST_PointN" en la página 483

ST_SymDifference

ST_SymDifference toma dos geometrías como parámetros de entrada y devuelve la geometría que es la diferencia simétrica de las dos geometrías. La diferencia simétrica es la parte que no forma intersección de las dos geometrías proporcionadas. La geometría resultante se representa según el sistema de referencia espacial de la primera geometría. La dimensión de la geometría devuelta es la misma que la de las geometrías de entrada. Ambas geometrías deben tener la misma dimensión.

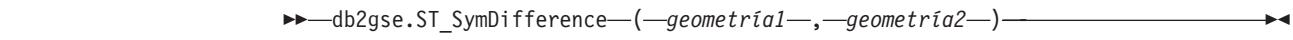
Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

Si las geometrías son iguales, se devuelve una geometría vacía de tipo ST_Point. Si cualquiera de las geometrías es un valor nulo, se devuelve un valor nulo.

La geometría resultante se representa según el tipo espacial más apropiado. Si se puede representar como punto, cadena lineal o polígono, se utiliza uno de estos tipos. En otro caso, se representa como multipunto, multilínea o multipolígono.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

geometría1
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la primera geometría para calcular la diferencia simétrica con *geometría2*.

geometría2
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la segunda geometría para calcular la diferencia simétrica con *geometría1*.

Restricciones para los datos geodésicos:

- Ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.
- ST_SymDifference sólo soporta los tipos de datos ST_Point, ST_Polygon, ST_MultiPoint y ST_MultiPolygon.

Tipo de retorno:

db2gse.ST_Geometry

Ejemplos:

Estos ejemplos muestran el uso de la función ST_SymDifference. Las geometrías están almacenadas en la tabla SAMPLE_GEOMS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms
VALUES( 1,
          ST_Geometry ('polygon ( (10 10, 10 20, 20 20, 20 10, 10 10) )', 0))

INSERT INTO sample_geoms
  VALUES
    (2, ST_Geometry ('polygon ( (30 30, 30 50, 50 50, 50 30, 30 30) )', 0))

INSERT INTO sample_geoms
  VALUES
    (3,ST_Geometry ('polygon ( (40 40, 40 60, 60 60, 60 40, 40 40) )', 0))

INSERT INTO sample_geoms
  VALUES
    (4, ST_Geometry ('linestring (70 70, 80 80)' , 0) )

INSERT INTO sample_geoms
  VALUES
    (5, ST_Geometry('linestring(75 75, 90 90)' ,0));
```

En los ejemplos siguientes, los resultados se han reformateado para mejorar la legibilidad. Los resultados dependerán de la pantalla utilizada.

Ejemplo 1:

Este ejemplo utiliza ST_SymDifference para obtener la diferencia simétrica de dos polígonos inconexos de la tabla SAMPLE_GEOMS.

```
SELECT a.id, b.id,
    CAST (ST_AsText (ST_SymDifference (a.geometry, b.geometry) )
    AS VARCHAR(350) ) SYM_DIFF
  FROM sample_geoms a, sample_geoms b
  WHERE a.id = 1 AND b.id = 2
```

Resultados:

```
ID   ID  SYM_DIFF
----- ----- ------------------------------------------------------------------
    1     2 MULTIPOLYGON ((( 10.00000000 10.00000000, 20.00000000 10.00000000,
              20.00000000 20.00000000, 10.00000000 20.00000000,
              10.00000000 10.00000000)),
                       (( 30.00000000 30.00000000, 50.00000000 30.00000000,
              50.00000000 50.00000000, 30.00000000 50.00000000,
             30.00000000 30.00000000)))
```

Ejemplo 2:

Este ejemplo utiliza ST_SymDifference para obtener la diferencia simétrica de dos polígonos que forman intersección de la tabla SAMPLE_GEOMS.

```
SELECT a.id, b.id,
    CAST (ST_AsText (ST_SymDifference (a.geometry, b.geometry) )
    AS VARCHAR(500) ) SYM_DIFF
  FROM sample_geoms a, sample_geoms b
  WHERE a.id = 2 AND b.id = 3
```

Resultados:

```
ID   ID  SYM_DIFF
--- --- ------------------------------------------------------------------
  2   3  MULTIPOLYGON ((( 40.00000000 50.00000000, 50.00000000 50.00000000,
           50.00000000 40.00000000, 60.00000000 40.00000000,
           60.00000000 60.00000000, 40.00000000 60.00000000,
           40.00000000 50.00000000)),
                       (( 30.00000000 30.00000000, 50.00000000 30.00000000,
           50.00000000 40.00000000, 40.00000000 40.00000000,
           40.00000000 50.00000000, 30.00000000 50.00000000,
           30.00000000 30.00000000)))
```

Ejemplo 3:

Este ejemplo utiliza ST_SymDifference para obtener la diferencia simétrica de dos cadenas lineales que forman intersección de la tabla SAMPLE_GEOMS.

```
SELECT a.id, b.id,
    CAST (ST_AsText (ST_SymDifference (a.geometry, b.geometry) )
    AS VARCHAR(350) ) SYM_DIFF
  FROM sample_geoms a, sample_geoms b
  WHERE a.id = 4 AND b.id = 5
```

Resultados:

```
ID   ID  SYM_DIFF
---- --- ------------------------------------------------------------------
   4   5 MULTILINESTRING (( 70.00000000 70.00000000, 75.00000000 75.00000000),
               ( 80.00000000 80.00000000, 90.00000000 90.00000000))
```

Información relacionada:

• "ST_Difference" en la página 377

ST_ToGeomColl

ST_ToGeomColl toma una geometría como parámetro de entrada y la convierte a un conjunto de geometrías. El conjunto de geometrías resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

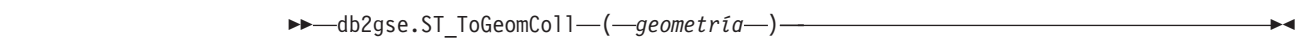
Si la geometría especificada está vacía, puede ser de cualquier tipo. Pero se convierte a ST_Multipoint, ST_MultiLineString o ST_MultiPolygon según convenga.

Si la geometría especificada no está vacía, debe ser de tipo ST_Point, ST_LineString o ST_Polygon. Luego, la geometría se convierte al tipo ST_Multipoint, ST_MultiLineString o ST_MultiPolygon respectivamente.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de los subtipos que representan la geometría que se convierte a un conjunto de geometrías.

Tipo de retorno:

db2gse.ST_GeomCollection

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de la función ST_ToGeomColl.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
  VALUES (1, ST_Polygon ('polygon ((3 3, 4 6, 5 3, 3 3))', 1)),
         (2, ST_Point ('point (1 2)', 1))
```

La sentencia SELECT siguiente se utiliza la función ST_ToGeomColl para obtener geometrías en forma de sus correspondientes subtipos de colección de geometrías.

```
SELECT id, CAST( ST_AsText( ST_ToGeomColl(geometry) )
    AS VARCHAR(120) ) GEOM_COLL
FROM sample_geometries
```

Resultados:

```
ID          GEOM_COLL
----------- ----------------------------------------------------------
          1 MULTIPOLYGON ((( 3.00000000 3.00000000, 5.00000000
                             3.00000000, 4.00000000 6.00000000,
                             3.00000000 3.00000000)))
          2 MULTIPOINT ( 1.00000000 2.00000000)
```

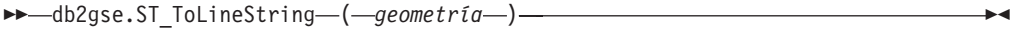
ST_ToLineString

ST_ToLineString toma una geometría como parámetro de entrada y la convierte en una cadena lineal. La cadena lineal resultante se representada según el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser una cadena lineal. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se convierte en una cadena lineal.

Una geometría se puede convertir en una cadena lineal si está vacía o si es una cadena lineal. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

Tipo de retorno:

db2gse.ST_LineString

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de la función ST_ToLineString.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
  VALUES (1, ST_Geometry ('linestring z (0 10 1, 0  0 3, 10 0 5)', 0)),
         (2, ST_Geometry ('point empty', 1) ),
         (3, ST_Geometry ('multipolygon empty', 1) )
```

La siguiente sentencia SELECT utiliza la función ST_ToLineString para obtener cadenas lineales que se han convertido al tipo ST_LineString a partir del tipo estático ST_Geometry.

```
SELECT CAST( ST_AsText( ST_ToLineString(geometry) )
     AS VARCHAR(130) ) LINES
FROM sample_geometries
```

Resultados:

```
LINES
------------------------------------------------------------------
LINESTRING Z ( 0.00000000 10.00000000 1.00000000, 0.00000000
               0.00000000 3.00000000, 10.00000000 0.00000000
               5.00000000)
LINESTRING EMPTY
LINESTRING EMPTY
```

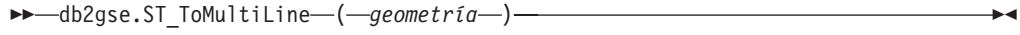
ST_ToMultiLine

ST_ToMultiLine toma una geometría como parámetro de entrada y la convierte en una multilínea. La multilínea resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser una multilínea o cadena lineal. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se convierte en una multilínea.

Una geometría se puede convertir en una multilínea si está vacía, o si es una cadena lineal o multilínea. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

Tipo de retorno:

db2gse.ST_MultiLineString

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de la función ST_ToMultiLine.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
  VALUES (1, ST_Geometry ('multilinestring ((0 10 1, 0 0 3, 10 0 5),
                                           (23 43, 27 34, 35 12))', 0) ),
         (2, ST_Geometry ('linestring z (0 10 1, 0 0 3, 10 0 5)', 0) ),
         (3, ST_Geometry ('point empty', 1) ),
         (4, ST_Geometry ('multipolygon empty', 1) )
```

La siguiente sentencia SELECT utiliza la función ST_ToMultiLine para obtener multilíneas que se han convertido al tipo ST_MultiLineString a partir del tipo estático ST_Geometry.

```
SELECT CAST( ST_AsText( ST_ToMultiLine(geometry) )
  AS VARCHAR(130) ) LINES
  FROM sample_geometries
```

Resultados:

```
LINES
-----------------------------------------------------
MULTILINESTRING Z ( 0.00000000 10.00000000 1.00000000,
                    0.00000000 0.00000000 3.00000000,
                   10.00000000 0.00000000 5.00000000)
MULTILINESTRING EMPTY
MULTILINESTRING EMPTY
```

ST_ToMultiPoint

ST_ToMultiPoint toma una geometría como parámetro de entrada y la convierte en multipunto. El multipunto resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser un punto o multipunto. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se convierte en un multipunto.

Una geometría se puede convertir en un multipunto si está vacía, si es un punto o si es un multipunto. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

Tipo de retorno:

db2gse.ST_MultiPoint

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de la función ST_ToMultiPoint.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
  VALUES (1, ST_Geometry ('multipoint (0 0, 0 4)', 1) ),
         (2, ST_Geometry ('point (30 40)', 1) ),
         (3, ST_Geometry ('multipolygon empty', 1) )
```

La siguiente sentencia SELECT utiliza la función ST_ToMultiPoint para obtener multipuntos que se han convertido al tipo ST_MultiPoint a partir del tipo estático ST_Geometry.

```
SELECT CAST( ST_AsText( ST_ToMultiPoint(geometry))
  AS VARCHAR(62) ) MULTIPOINTS
FROM sample_geometries
```

Resultados:

```
MULTIPOINTS
------------------------------------------------------------
MULTIPOINT ( 0.00000000 0.00000000, 0.00000000 4.00000000)
MULTIPOINT ( 30.00000000 40.00000000)
MULTIPOINT EMPTY
```

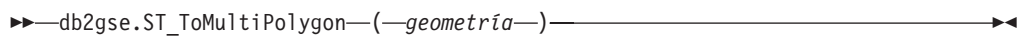
ST_ToMultiPolygon

ST_ToMultiPolygon toma una geometría como parámetro de entrada y la convierte en un multipolígono. El multipolígono resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser un polígono o multipolígono. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se convierte en un multipolígono.

Una geometría se puede convertir en un multipolígono si está vacía, si es un polígono o un multipolígono. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

Tipo de retorno:

db2gse.ST_MultiPolygon

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo crea varias geometrías y luego utiliza ST_ToMultiPolygon para obtener multipolígonos.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
  VALUES (1, ST_Geometry ('polygon ((0 0, 0 4, 5 4, 5 0, 0 0))', 1)),
         (2, ST_Geometry ('point empty', 1)),
         (3, ST_Geometry ('multipoint empty', 1))
```

La siguiente sentencia SELECT utiliza la función ST_ToMultiPolygon para obtener multipolígonos que se han convertido al tipo ST_MultiPolygon a partir del tipo estático ST_Geometry.

```
SELECT CAST( ST_AsText( ST_ToMultiPolygon(geometry) )
  AS VARCHAR(130) ) POLYGONS
  FROM sample_geometries
```

Resultados:

```
POLYGONS
------------------------------------------------------------------------
MULTIPOLYGON (( 0.00000000 0.00000000, 5.00000000 0.00000000,
                5.00000000 4.00000000, 0.00000000 4.00000000,
                 0.00000000 0.00000000))

MULTIPOLYGON EMPTY

MULTIPOLYGON EMPTY
```

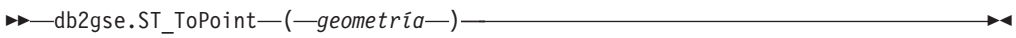
ST_ToPoint

ST_ToPoint toma una geometría como parámetro de entrada y la convierte en un punto. El punto resultante se representa utilizando el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser un punto. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetro:

geometría
Valor de tipo ST_Geometry o uno de los subtipos que representa la geometría que se convierte a punto.

Una geometría se puede convertir a un punto si está vacía o es un punto. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

Tipo de retorno:

db2gse.ST_Point

Ejemplo:

Este ejemplo crea tres geometrías en SAMPLE_GEOMETRIES y convierte cada geometría en un punto.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
  VALUES (1, ST_Geometry ('point (30 40)', 1) ),
         (2, ST_Geometry ('linestring empty', 1) ),
         (3, ST_Geometry ('multipolygon empty', 1) )
```

La siguiente sentencia SELECT utiliza la función ST_ToPoint para obtener puntos que se han convertido al tipo ST_Point a partir del tipo estático ST_Geometry.

```
SELECT CAST( ST_AsText( ST_ToPoint(geometry) ) AS VARCHAR(35) ) POINTS
  FROM sample_geometries
```

Resultados:

```
POINTS
---------------------------------
POINT ( 30.00000000 40.00000000)
POINT EMPTY
POINT EMPTY
```

ST_ToPolygon

ST_ToPolygon toma una geometría como parámetro de entrada y la convierte en un polígono. El polígono resultante se representa según el sistema de referencia espacial de la geometría proporcionada.

La geometría proporcionada debe estar vacía o ser un polígono. Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:

Parámetro:

geometría
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se convierte a un polígono.

Una geometría se puede convertir a un polígono si está vacía o es un polígono. Si no se puede realizar la conversión, se emite una condición de excepción (SQLSTATE 38SUD).

Tipo de retorno:

db2gse.ST_Polygon

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo crea tres geometrías en SAMPLE_GEOMETRIES y convierte cada geometría en un polígono.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
  VALUES (1, ST_Geometry ('polygon ((0 0, 0 4, 5 4, 5 0, 0 0))', 1) ),
         (2, ST_Geometry ('point empty', 1) ),
         (3, ST_Geometry ('multipolygon empty', 1) )
```

La siguiente sentencia SELECT utiliza la función ST_ToPolygon para obtener polígonos que se han convertido al tipo ST_Polygon a partir del tipo estático ST_Geometry.

```
SELECT CAST( ST_AsText( ST_ToPolygon(geometry) ) AS VARCHAR(130) ) POLYGONS
  FROM sample_geometries
```

Resultados:

```
POLYGONS
--------------------------------------------------------------------------------
POLYGON (( 0.00000000 0.00000000, 5.00000000 0.00000000,
           5.00000000 4.00000000,0.00000000 4.00000000,
            0.00000000 0.00000000))

POLYGON EMPTY

POLYGON EMPTY
```

ST_Touches

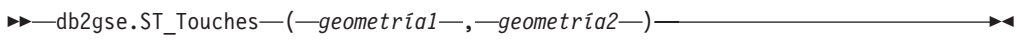
ST_Touches toma dos geometrías como parámetros de entrada y devuelve un 1 si las geometrías proporcionadas están espacialmente en contacto. En caso contrario, se devuelve un 0 (cero).

Dos geometrías están en contacto si los interiores de ambas geometrías no forman intersección, pero el perímetro de una de ellas forma intersección con el perímetro o el interior de la otra geometría.

Si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial.

Si ambas geometrías proporcionadas son puntos o multipuntos, o si cualquiera de las dos geometrías es un valor nulo o está vacía, se devuelve un valor nulo.

Sintaxis:



Parámetros:

geometría1
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la cual se debe verificar si está en contacto con *geometría2*.

geometría2
Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría para la cual se debe verificar si está en contacto con *geometría1*.

Tipo de retorno:

INTEGER

Ejemplo:

Este ejemplo añade varias geometrías a la tabla SAMPLE_GEOMS. Luego, se utiliza la función ST_Touches para determinar qué geometrías están en contacto entre sí.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms
  VALUES (1, ST_Geometry('polygon ( (20 30, 30 30, 30 40, 20 40, 20 30) )' , 0) )

INSERT INTO sample_geoms
   VALUES (2, ST_Geometry('polygon ( (30 30, 30 50, 50 50, 50 30, 30 30) )' ,0) )

INSERT INTO sample_geoms
  VALUES (3, ST_Geometry('polygon ( (40 40, 40 60, 60 60, 60 40, 40 40) )' , 0) )

INSERT INTO sample_geoms
  VALUES (4, ST_Geometry('linestring( 60 60, 70 70 )' , 0) )

INSERT INTO sample_geoms
  VALUES (5, ST_Geometry('linestring( 30 30, 60 60 )' , 0) )

SELECT a.id, b.id, ST_Touches (a.geometry, b.geometry) TOUCHES
  FROM sample_geoms a, sample_geoms b
  WHERE b.id >= a.id
```

Resultados:

```
     ID          ID          TOUCHES
     ----------- ----------- -----------
               1           1           0
               1           2           1
               1           3           0
               1           4           0
               1           5           1
               2           2           0
               2           3           0
               2           4           0
               2           5           1
               3           3           0
               3           4           1
               3           5           1
               4           4           0
               4           5           1
               5           5           0
```

Información relacionada:

- "Funciones que utilizan índices para optimizar consultas" en la página 128

ST_Transform

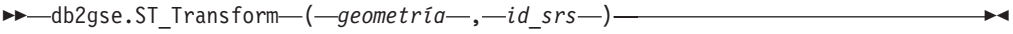
ST_Transform toma como parámetros de entrada una geometría y un identificador de sistemas de referencia espacial y transforma la geometría para que esté representada en el sistema de referencia espacial especificado. Se realizan proyecciones y conversiones entre diferentes sistemas de coordenadas y se ajustan las coordenadas de las geometrías según convenga.

La geometría se puede convertir al sistema de referencia espacial especificado sólo si éste está basado en el mismo sistema de coordenadas geográficas que el sistema de referencia espacial actual de la geometría. Si el sistema de referencia espacial especificado o el utilizado actualmente por la geometría está basado en un sistema de coordenadas proyectadas, se realiza una proyección inversa para determinar el sistema de coordenadas geográficas en el que se basa el sistema proyectado.

Si la geometría proporcionada es un valor nulo, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

geometría
: Valor de tipo ST_Geometry o uno de sus subtipos que representa la geometría que se transforma en el sistema de referencia espacial identificado por *id_srs*.

id_srs
: Valor de tipo INTEGER que identifica el sistema de referencia espacial correspondiente a la geometría resultante.

  Si no puede realizarse la conversión al sistema de referencia espacial especificado debido a que el utilizado actualmente por *geometría* no es compatible con el especificado por *id_srs*, se emite una condición de excepción (SQLSTATE 38SUC).

  Si *id_srs* no identifica uno de los sistemas de referencia espacial listados en la vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, se emite una condición de excepción (SQLSTATE 38SU1).

Tipo de retorno:

db2gse.ST_Geometry

Ejemplos:

Los ejemplos siguientes muestran el uso de ST_Transform para convertir una geometría desde un sistema de referencia espacial a otro.

Primero, se invoca a db2se para crear un sistema de referencia espacial de plano de estado cuyo ID sea 3.

```
db2se create_srs SAMP_DB
   -srsId 3 -srsName z3101a -xOffset 0 -yOffset 0 -xScale 1 -yScale 1
   - coordsysName NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet
```

A continuación se añaden puntos a:

- La tabla SAMPLE_POINTS_SP con coordenadas de plano de estado utilizando ese sistema de referencia espacial.
- la tabla SAMPLE_POINTS_LL utilizando coordenadas especificadas de latitud y longitud.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points_sp (id INTEGER, geometry ST_Point)
CREATE TABLE sample_points_ll (id INTEGER, geometry ST_Point)

INSERT INTO sample_points_sp
  VALUES (12457, ST_Point('point ( 567176.0 1166411.0)', 3) )

INSERT INTO sample_points_sp
  VALUES (12477, ST_Point('point ( 637948.0 1177640.0)', 3) )

INSERT INTO sample_points_ll
  VALUES (12457, ST_Point('point ( -74.22371600 42.03498700)', 1) )

INSERT INTO sample_points_ll
  VALUES (12477, ST_Point('point ( -73.96293200 42.06487900)', 1) )
```

Luego, se utiliza la función ST_Transform para convertir las geometrías.

Ejemplo 1:

Este ejemplo convierte puntos con coordenadas de latitud y longitud en coordenadas de plano de estado.

```
SELECT id, CAST( ST_AsText( ST_Transform( geometry, 3) )
    AS VARCHAR(100) ) STATE_PLANE
  FROM sample_points_ll
```

Resultados:

```
ID          STATE_PLANE
----------- ----------------------------------------
      12457 POINT ( 567176.00000000 1166411.00000000)
      12477 POINT ( 637948.00000000 1177640.00000000)
```

Ejemplo 2:

Este ejemplo convierte puntos con coordenadas de plano de estado en coordenadas de latitud y longitud.

```
SELECT id, CAST( ST_AsText( ST_Transform( geometry, 1) )
    AS VARCHAR(100) ) LAT_LONG
  FROM sample_points_sp
```

Resultados:

```
ID          LAT_LONG
----------- ----------------------------------------------------------------
      12457 POINT ( -74.22371500 42.03498800)
      12477 POINT ( -73.96293100 42.06488000)
```

Información relacionada:

- "Vista de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS" en la página 303

ST_Union

ST_Union toma dos geometrías como parámetros de entrada y devuelve la geometría que es la unión de las geometrías proporcionadas. La geometría resultante se representa según el sistema de referencia espacial de la primera geometría.

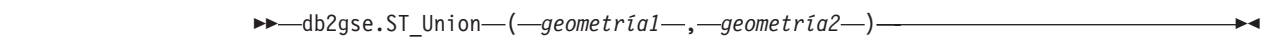
Ambas geometrías deben tener la misma dimensión. Si cualquiera de las dos geometrías proporcionadas es un valor nulo, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

La geometría resultante se representa según el tipo espacial más apropiado. Si se puede representar como punto, cadena lineal o polígono, se utiliza uno de estos tipos. En otro caso, se representa como multipunto, multilínea o multipolígono.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

geometría1
Valor de tipo ST_Geometry o uno de sus subtipos que se combina con *geometría2*.

geometría2
Valor de tipo ST_Geometry o uno de sus subtipos que se combina con *geometría1*.

Restricciones para los datos geodésicos: Ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

Tipo de retorno:

db2gse.ST_Geometry

Ejemplos:

Las siguientes sentencias de SQL crean y llenan la tabla SAMPLE_GEOMS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry, ST_Geometry)

INSERT INTO sample_geoms
  VALUES (1, ST_Geometry( 'polygon
         ((10 10, 10 20, 20 20, 20 10, 10 10) )', 0))

INSERT INTO sample_geoms
  VALUES (2, ST_Geometry( 'polygon
         ((30 30, 30 50, 50 50, 50 30, 30 30) )', 0))
```

```
INSERT INTO sample_geoms
  VALUES (3, ST_Geometry( 'polygon
         ((40 40, 40 60, 60 60, 60 40, 40 40) )', 0))

INSERT INTO sample_geoms
  VALUES (4, ST_Geometry('linestring (70 70, 80 80)', 0))

INSERT INTO sample_geoms
  VALUES (5, ST_Geometry('linestring (80 80, 100 70)', 0))
```

En los ejemplos siguientes, los resultados se han reformateado para mejorar la legibilidad. Los resultados dependerán de la pantalla utilizada.

Ejemplo 1:

Este ejemplo obtiene la unión de dos polígonos inconexos.

```
SELECT a.id, b.id, CAST ( ST_AsText( ST_Union( a.geometry, b.geometry) )
    AS VARCHAR (350) ) UNION
  FROM sample_geoms a, sample_geoms b
  WHERE a.id = 1 AND b.id = 2
```

Resultados:

```
ID    ID    UNION
----- ----- ------------------------------------------------------------
    1     2 MULTIPOLYGON ((( 10.00000000 10.00000000, 20.00000000
                    10.00000000, 20.00000000 20.00000000, 10.00000000
                    20.00000000, 10.00000000 10.00000000))
                            (( 30.00000000 30.00000000, 50.00000000
                    30.00000000,50.00000000 50.00000000, 30.00000000
                    50.00000000,30.00000000 30.00000000)))
```

Ejemplo 2:

Este ejemplo obtiene la unión de dos polígonos que forman intersección.

```
SELECT a.id, b.id, CAST ( ST_AsText( ST_Union(a.geometry, b.geometry))
    AS VARCHAR (250)) UNION
  FROM sample_geoms a, sample_geoms b
  WHERE a.id = 2 AND b.id = 3
```

Resultados:

```
ID    ID    UNION
----- ----- ----------------------------------------------------
    2     3 POLYGON (( 30.00000000 30.00000000, 50.00000000
                30.00000000,50.00000000 40.00000000, 60.00000000
                40.00000000,60.00000000 60.00000000, 40.00000000
                60.00000000 40.00000000 50.00000000, 30.00000000
                50.00000000, 30.00000000 30.00000000))
```

Ejemplo 3:

Este ejemplo obtiene la unión de dos cadenas lineales.

```
SELECT a.id, b.id, CAST ( ST_AsText( ST_Union( a.geometry, b.geometry) )
    AS VARCHAR (250) ) UNION
  FROM sample_geoms a, sample_geoms b
  WHERE a.id = 4 AND b.id = 5
```

Resultados:

```
ID    ID    UNION
----- ----- ------------------------------------------------------------
    4     5 MULTILINESTRING (( 70.00000000 70.00000000, 80.00000000 80.00000000),
                ( 80.00000000 80.00000000, 100.00000000 70.00000000))
```

ST_Within

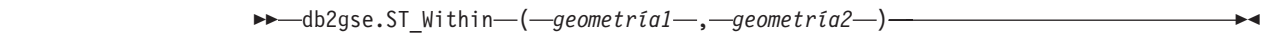
ST_Within toma dos geometrías como parámetros de entrada y devuelve un 1 si la primera geometría está completamente dentro de la segunda. En caso contrario, se devuelve un 0 (cero).

Si cualquiera de las geometrías proporcionadas es un valor nulo o está vacía, se devuelve un valor nulo.

Para los datos no geodésicos, si la segunda geometría no está representada en el mismo sistema de referencia espacial que la primera geometría, se convertirá al otro sistema de referencia espacial. Para los datos geodésicos, ambas geometrías deben estar en el mismo sistema de referencia espacial (SRS) geodésico.

ST_Within efectúa la misma operación lógica que ST_Contains, con los parámetros invertidos.

Sintaxis:



Parámetros:

geometría1
Valor de tipo ST_Geometry o uno de sus subtipos para el que se comprobar si está completamente dentro de *geometría2*.

geometría2
Valor de tipo ST_Geometry o uno de sus subtipos para el que se comprobar si está completamente dentro de *geometría1*.

Restricciones para los datos geodésicos: Ambas geometrías deben ser geodésicas y deben estar en el mismo SRS geodésico.

Tipo de retorno:

INTEGER

Ejemplos:

Estos ejemplos muestran el uso de la función ST_Within. Se crean e insertan geometrías en tres tablas: SAMPLE_POINTS, SAMPLE_LINES y SAMPLE_POLYGONS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)
CREATE TABLE sample_polygons (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_points (id, geometry)
  VALUES (1, ST_Point (10, 20, 1) ),
         (2, ST_Point ('point (41 41)', 1) )

INSERT INTO sample_lines (id, line)
  VALUES (10, ST_LineString ('linestring (1 10, 3 12, 10 10)', 1) ),
         (20, ST_LineString ('linestring (50 10, 50 12, 45 10)', 1) )

INSERT INTO sample_polygons (id, geometry)
  VALUES (100, ST_Polygon ('polygon (( 0 0, 0 40, 40 40, 40 0, 0 0))', 1) )
```

Ejemplo 1:

Este ejemplo obtiene puntos de la tabla SAMPLE_POINTS que están en polígonos de la tabla SAMPLE_POLYGONS.

```
SELECT a.id POINT_ID_WITHIN_POLYGONS
  FROM sample_points a, sample_polygons b
  WHERE ST_Within( b.geometry, a.geometry) = 0
```

Resultados:

```
POINT_ID_WITHIN_POLYGONS
----------------------
2
```

Ejemplo 2:

Este ejemplo obtiene cadenas lineales de la tabla SAMPLE_LINES que están en polígonos de la tabla SAMPLE_POLYGONS.

```
SELECT a.id LINE_ID_WITHIN_POLYGONS
  FROM sample_lines a, sample_polygons b
  WHERE ST_Within( b.geometry, a.geometry) = 0
```

Resultados:

```
LINE_ID_WITHIN_POLYGONS
----------------------
                     1
```

Información relacionada:

- "ST_Contains" en la página 372

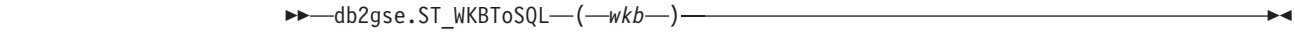
ST_WKBToSQL

ST_WKBToSQL toma una representación binaria convencional de una geometría y devuelve la geometría correspondiente. El sistema de referencia espacial con el identificador 0 (cero) se utiliza para la geometría resultante.

Si la representación binaria convencional proporcionada es un valor nulo, se devuelve un valor nulo.

ST_WKBToSQL(*wkb*) produce el mismo resultado que ST_Geometry(*wkb* ,0). Debido a su flexibilidad, es recomendable utilizar la función ST_Geometry en lugar de ST_WKBToSQL: ST_Geometry utiliza otras modalidades de datos de entrada además de la representación binaria convencional.

Sintaxis:



Parámetro:

wkb Valor de tipo BLOB(2G) que contiene la representación binaria convencional de la geometría resultante.

Tipo de retorno:

db2gse.ST_Geometry

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de la función ST_WKBToSQL. Primero, se almacenan geometrías en la columna GEOMETRY de la tabla SAMPLE_GEOMETRIES. Luego, sus representaciones binarias convencionales se guardan en la columna WKB utilizando la función ST_AsBinary en la sentencia UPDATE. Finalmente, se utiliza la función ST_WKBToSQL para obtener las coordenadas de las geometrías contenidas en la columna WKB.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries
  (id INTEGER, geometry ST_Geometry, wkb BLOB(32K) )

INSERT INTO sample_geometries (id, geometry)
  VALUES (10, ST_Point ( 'point (44 14)', 0 ) ),
         (11, ST_Point ( 'point (24 13)', 0 ) ),
         (12, ST_Polygon ('polygon ((50 20, 50 40, 70 30, 50 20))', 0 ) )
UPDATE sample_geometries AS temp_correlated
  SET wkb = ST_AsBinary(geometry)
  WHERE id = temp_correlated.id
```

Utilice esta sentencia SELECT para ver las geometrías contenidas en la columna WKB.

```
SELECT id, CAST( ST_AsText( ST_WKBToSQL(wkb) ) AS VARCHAR(120) ) GEOMETRIES
  FROM sample_geometries
```

Resultados:

```
ID          GEOMETRIES
----------- ------------------------------------------------------------
         10 POINT ( 44.00000000 14.00000000)
         11 POINT ( 24.00000000 13.00000000)
         12 POLYGON (( 50.00000000 20.00000000, 70.00000000 30.00000000,
                       50.00000000 40.00000000, 50.00000000 20.00000000))
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_Geometry" en la página 406
- "ST_WKTToSQL" en la página 514

ST_WKTToSQL

ST_WKTToSQL toma una representación de texto convencional de una geometría y devuelve la geometría correspondiente. El sistema de referencia espacial con el identificador 0 (cero) se utiliza para la geometría resultante.

Si la representación de texto convencional proporcionada es un valor nulo, se devuelve un valor nulo.

ST_WKTToSQL(*wkt*) produce el mismo resultado que ST_Geometry(*wkt* ,0). Debido a su flexibilidad, es recomendable utilizar la función ST_Geometry en lugar de ST_WKTToSQL: ST_Geometry utiliza otras modalidades de datos de entrada además de la representación de texto convencional.

Sintaxis:

Parámetro:

wkt Valor de tipo CLOB(2G) que contiene la representación de texto convencional de la geometría resultante.

Tipo de retorno:

db2gse.ST_Geometry

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de ST_WKTToSQL para crear e insertar geometrías utilizando sus representaciones de texto convencionales.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
  VALUES (10, ST_WKTToSQL( 'point (44 14)' ) ),
         (11, ST_WKTTSQL ( 'point (24 13)' ) ),
         (12, ST_WKTToSQL ('polygon ((50 20, 50 40, 70 30, 50 20))' ) )
```

Esta sentencia SELECT obtiene las geometrías que se han insertado.

```
SELECT id, CAST( ST_AsText(geometry) AS VARCHAR(120) ) GEOMETRIES
  FROM sample_geometries
```

Resultados:

```
ID          GEOMETRIES
----------- ------------------------------------------------------------
         10 POINT ( 44.00000000 14.00000000)
         11 POINT ( 24.00000000 13.00000000)
         12 POLYGON (( 50.00000000 20.00000000, 70.00000000 30.00000000,
                       50.00000000 40.00000000, 50.00000000 20.00000000))
```

Conceptos relacionados:

- "Datos espaciales y geodésicos" en la página 4

Información relacionada:

- "ST_Geometry" en la página 406
- "ST_WKBToSQL" en la página 513

ST_X

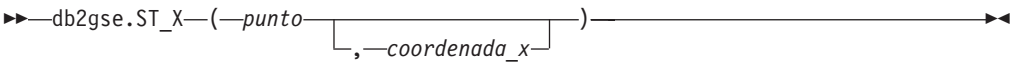
ST_X puede actuar de una de estas maneras:

- Toma un punto como parámetro de entrada y devuelve su coordenada X
- Toma un punto y una coordenada X como parámetros de entrada y devuelve el propio punto con su coordenada X establecida en el valor proporcionado.

Si el punto proporcionado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

punto Valor de tipo ST_Point para el cual se obtiene o modifica la coordenada X.

coordenada_x
Valor de tipo DOUBLE que representa la nueva coordenada X del *punto*.

Tipos de retorno:

- DOUBLE, si no se especifica *coordenada_x*
- db2gse.ST_Point, si se especifica *coordenada_x*

Ejemplos:

Estos ejemplos muestran el uso de la función ST_X. Se crean e insertan geometrías en la tabla SAMPLE_POINTS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points (id, geometry)
  VALUES (1, ST_Point (2, 3, 32, 5, 1) ),
         (2, ST_Point (4, 5, 20, 4, 1) ),
         (3, ST_Point (3, 8, 23, 7, 1) )
```

Ejemplo 1:

Este ejemplo determina las coordenadas X de los puntos contenidos en la tabla.

```
SELECT id, ST_X (geometry) X_COORD
  FROM sample_points
```

Resultados:

```
ID         X_COORD
---------- ----------------------
         1 +2.00000000000000E+000
         2 +4.00000000000000E+000
         3 +3.00000000000000E+000
```

Ejemplo 2:

Este ejemplo devuelve un punto con su coordenada X establecida en el valor 40.

```
SELECT id, CAST( ST_AsText( ST_X (geometry, 40)) AS VARCHAR(60) )
  X_40
  FROM sample_points
  WHERE id=3
```

Resultados:

```
ID         X_40
------ ------------------------------------------------------------
     3 POINT ZM ( 40.00000000 8.00000000 23.00000000 7.00000000)
```

Información relacionada:
- “ST_M” en la página 434
- “ST_Y” en la página 517
- “ST_Z” en la página 518

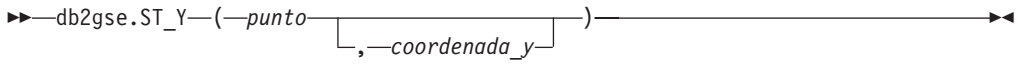
ST_Y

ST_Y puede actuar de una de estas maneras:
- Toma un punto como parámetro de entrada y devuelve su coordenada Y
- Toma un punto y una coordenada Y como parámetros de entrada y devuelve el propio punto con su coordenada Y establecida en el valor proporcionado

Si el punto proporcionado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

punto Valor de tipo ST_Point para el cual se obtiene o modifica la coordenada Y.

coordenada_y
Valor de tipo DOUBLE que representa la nueva coordenada Y del *punto*.

Tipos de retorno:
- DOUBLE, si no se especifica *coordenada_y*
- db2gse.ST_Point, si se especifica *coordenada_y*

Ejemplos:

Estos ejemplos muestran el uso de la función ST_Y. Se crean e insertan geometrías en la tabla SAMPLE_POINTS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points (id, geometry)
  VALUES (1, ST_Point (2, 3, 32, 5, 1) ),
         (2, ST_Point (4, 5, 20, 4, 1) ),
         (3, ST_Point (3, 8, 23, 7, 1) )
```

Ejemplo 1:

Este ejemplo determina las coordenadas Y de los puntos contenidos en la tabla.

```
SELECT id, ST_Y (geometry) Y_COORD
  FROM sample_points
```

Resultados:

```
ID         Y_COORD
---------- ----------------------
         1 +3.00000000000000E+000
         2 +5.00000000000000E+000
         3 +8.00000000000000E+000
```

Ejemplo 2:

Este ejemplo devuelve un punto con su coordenada Y establecida en el valor 40.

```
SELECT id, CAST( ST_AsText( ST_Y (geometry, 40)) AS VARCHAR(60) )
  Y_40
  FROM sample_points
  WHERE id=3
```

Resultados:

```
ID         Y_40
-------- ------------------------------------------------------
     3 POINT ZM ( 3.00000000 40.00000000 23.00000000 7.00000000)
```

Información relacionada:

- "ST_M" en la página 434
- "ST_X" en la página 515
- "ST_Z" en la página 518

ST_Z

ST_Z puede actuar de una de estas maneras:

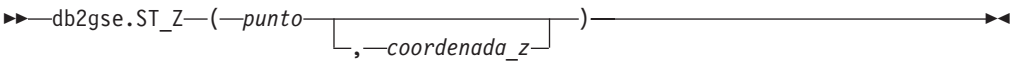
- Toma un punto como parámetro de entrada y devuelve su coordenada Z
- Toma como parámetros de entrada un punto y una coordenada Z y devuelve el propio punto junto con su coordenada Z establecida en el valor proporcionado, aunque el punto especificado no tenga ninguna coordenada Z.

Si la coordenada Z especificada es nula, la coordenada Z se elimina del punto.

Si el punto especificado es un valor nulo o está vacío, se devuelve un valor nulo.

También se puede invocar a esta función como método.

Sintaxis:



Parámetros:

punto Valor de tipo ST_Point para el cual se obtiene o modifica la coordenada Z.

coordenada_z
Valor de tipo DOUBLE que representa la nueva coordenada Z del *punto*.

Si *coordenada_z* es nulo, la coordenada Z se elimina de *punto*.

Tipos de retorno:

- DOUBLE, si no se especifica *coordenada_z*
- db2gse.ST_Point, si se especifica *coordenada_z*

Ejemplos:

Estos ejemplos muestran el uso de la función ST_Z. Se crean e insertan geometrías en la tabla SAMPLE_POINTS.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points (id, geometry)
  VALUES (1, ST_Point (2, 3, 32, 5, 1) ),
         (2, ST_Point (4, 5, 20, 4, 1) ),
         (3, ST_Point (3, 8, 23, 7, 1) )
```

Ejemplo 1:

Este ejemplo determina las coordenadas Z de los puntos contenidos en la tabla.

```
SELECT id, ST_Z (geometry) Z_COORD
  FROM sample_points
```

Resultados:

```
ID         Z_COORD
---------- ----------------------
         1 +3.20000000000000E+001
         2 +2.00000000000000E+001
         3 +2.30000000000000E+001
```

Ejemplo 2:

Este ejemplo devuelve un punto con su coordenada Z establecida en el valor 40.

```
SELECT id, CAST( ST_AsText( ST_Z (geometry, 40)) AS VARCHAR(60) )
  Z_40
  FROM sample_points
  WHERE id=3
```

Resultados:

```
ID         Z_40
------ ------------------------------------------------------
     3 POINT ZM ( 3.00000000 8.00000000 40.00000000 7.00000000)
```

Información relacionada:

- "ST_M" en la página 434
- "ST_X" en la página 515
- "ST_Y" en la página 517

Union aggregate (Agregado de unión)

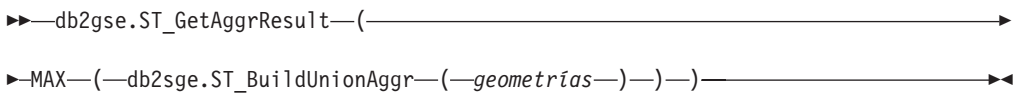
Un agregado de unión es la combinación de las funciones ST_BuildUnionAggr y ST_GetAggrResult. Esta combinación agrega una columna de geometrías de una tabla a una geometría individual mediante la creación de la unión.

Si todas las geometrías que se deben combinar en la unión son nulas, se devuelve un valor nulo. Si cualquiera de las geometrías que se deben combinar en la unión son nulas o están vacías, se devuelve una geometría vacía de tipo ST_Point.

La función ST_BuildUnionAggr también se puede invocar como método.

Sintaxis:

Union aggregate (Agregado de unión)



Parámetros:

geometrías
Columna de una tabla cuyo tipo es ST_Geometry o uno de sus subtipos que representa todas las geometrías que se deben combinar en una unión.

Tipo de retorno:

db2gse.ST_Geometry

Restricciones:

No puede crear el agregado de unión de una columna espacial de una tabla en los casos siguientes:

- En los entornos de proceso paralelo en masa (Massively Parallel Processing, MPP)
- Si se utiliza una cláusula GROUP BY en la sentencia SELECT
- Si utiliza una función distinta de la función de agregación MAX de DB2

Ejemplo:

En el ejemplo siguiente, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido dependerá de la pantalla utilizada.

Este ejemplo muestra el uso de un agregado de unión para combinar un conjunto de puntos y formar multipuntos. Primero se añaden varios puntos a la tabla SAMPLE_POINTS. Luego se utilizan las funciones ST_GetAggrResult y ST_BuildUnionAggr para crear la unión de los puntos.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points
  VALUES (1, ST_Point (2, 3, 1) )
INSERT INTO sample_points
  VALUES (2, ST_Point (4, 5, 1) )
INSERT INTO sample_points
  VALUES (3, ST_Point (13, 15, 1) )
INSERT INTO sample_points
  VALUES (4, ST_Point (12, 5, 1) )
INSERT INTO sample_points
  VALUES (5, ST_Point (23, 2, 1) )
INSERT INTO sample_points
  VALUES (6, ST_Point (11, 4, 1) )

SELECT CAST (ST_AsText(
          ST_GetAggrResult( MAX( ST_BuildUnionAggregate (geometry) ) ))
       AS VARCHAR(160)) POINT_AGGREGATE
  FROM sample_points
```

Resultados:

```
POINT_AGGREGATE
-------------------------------------------------------
MULTIPOINT ( 2.00000000 3.00000000, 4.00000000 5.00000000,
            11.00000000 4.00000000, 12.00000000 5.00000000,
            13.00000000 15.00000000, 23.00000000 2.00000000)
```

Union aggregate (Agregado de unión)

Capítulo 24. Grupos de transformación

Grupos de transformación

Spatial Extender proporciona cuatro grupos de transformación que se utilizan para transferir geometrías entre el servidor DB2 y una aplicación cliente. Estos grupos de transformación incluyen los siguientes formatos de intercambio de datos:

- Representación de texto convencional (WKT)
- Representación binaria convencional (WKB)
- Representación de forma ESRI
- Geography Markup Language (GML)

Cuando se recuperan datos de una tabla que contiene una columna espacial, los datos de la columna espacial se transforman en un valor de tipo CLOB(2G) o BLOB(2G), según si se ha indicado que los datos transformados se deben representar en formato binario o de texto. También puede utilizar los grupos de transformación para transferir datos espaciales a la base de datos.

Para seleccionar qué grupo de transformación se debe utilizar al transferir datos, utilice la sentencia SET CURRENT DEFAULT TRANSFORM GROUP para modificar el registro especial CURRENT DEFAULT TRANSFORM GROUP de DB2. DB2 utiliza el valor de este registro especial para determinar qué funciones de transformación se deben invocar para realizar las conversiones necesarias.

Los grupos de transformación pueden simplificar la programación de aplicaciones. En lugar de utilizar explícitamente funciones de conversión en las sentencias de SQL, puede especificar un grupo de transformación y dejar que DB2 se ocupe de la tarea.

Conceptos relacionados:

- "Grupo de transformación ST_WellKnownText" en la página 523
- "Grupo de transformación ST_WellKnownBinary" en la página 525
- "Grupo de transformación ST_Shape" en la página 526
- "Grupo de transformación ST_GML" en la página 527

Grupo de transformación ST_WellKnownText

Puede utilizar el grupo de transformación ST_WellKnownText para intercambiar datos con DB2® utilizando la representación de texto convencional (representación WKT).

Cuando se vincula un valor del servidor de bases de datos con el cliente, se utiliza la misma función proporcionada por ST_AsText() para convertir una geometría a la representación WKT. Cuando se transfiere la representación de texto convencional de una geometría al servidor de bases de datos, se utiliza implícitamente la función ST_Geometry(CLOB) para realizar las conversiones a un valor de tipo ST_Geometry. El uso del grupo de transformación para vincular valores con DB2 hace que las geometrías se representen en el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido variará según la pantalla que se utilice.

Ejemplo 1:

El siguiente script de SQL muestra cómo utilizar el grupo de transformación ST_WellKnownText para obtener una geometría en su representación de texto convencional sin utilizar la función explícita ST_AsText.

```
CREATE TABLE transforms_sample (
   id   INTEGER,
   geom  db2gse.ST_Geometry)

     INSERT
  INTO  transforms_sample
   VALUES (1, db2gse.ST_LineString('linestring
          (100 100, 200 100)', 0))

SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownText

   SELECT id, geom
   FROM transforms_sample
  WHERE  id = 1
```

Resultados:

```
ID   GEOM
---  ------------------------------------------------------------------
  1  LINESTRING ( 100.00000000 100.00000000, 200.00000000 100.00000000)
```

Ejemplo 2:

El siguiente código en C muestra cómo utilizar el grupo de transformación ST_WellKnownText para insertar geometrías utilizando la función explícita ST_Geometry para la variable de lenguaje principal `wkt_buffer`, cuyo tipo es CLOB y que contiene la representación de texto convencional del punto (10 10) que se debe insertar.

```
             EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS CLOB(1000) wkt_buffer;
 EXEC SQL END DECLARE SECTION;

// definir el grupo de transformación para todas las sentencias de SQL
// posteriores
 EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownText;

id = 100;
strcpy(wkt_buffer.data, "point ( 10 10 )");
wkt_buffer.length = strlen(wkt_buffer.data);

// insertar punto con WKT en la columna de tipo ST_Geometry
 EXEC SQL
     INSERT
    INTO   transforms_sample(id, geom)
    VALUES (:id, :wkt_buffer);
```

Grupo de transformación ST_WellKnownBinary

Utilice el grupo de transformación ST_WellKnownBinary para intercambiar datos con DB2® utilizando la representación binaria convencional (representación WKB).

Cuando se vincula un valor del servidor de bases de datos con el cliente, se utiliza la misma función proporcionada por ST_AsBinary() para convertir una geometría a la representación WKB. Cuando se transfiere la representación binaria convencional de una geometría al servidor de bases de datos, se utiliza implícitamente la función ST_Geometry(BLOB) para realizar las conversiones a un valor de tipo ST_Geometry. El uso del grupo de transformación para vincular valores con DB2 hace que las geometrías se representen en el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Ejemplo:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido variará según la pantalla que se utilice.

Ejemplo 1:

El siguiente script de SQL muestra cómo utilizar el grupo de transformación ST_WellKnownBinary para obtener una geometría en su representación binaria convencional sin utilizar la función explícita ST_AsBinary.

```
CREATE TABLE transforms_sample (
   id   INTEGER,
   geom  db2gse.ST_Geometry)

     INSERT
   INTO transforms_sample
   VALUES ( 1, db2gse.ST_Polygon('polygon ((10 10, 20 10, 20 20,
             10 20, 10 10))', 0))

SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownBinary

   SELECT id, geom
   FROM transforms_sample
   WHERE id = 1
```

Resultados:

```
ID     GEOM
----   ------------------------------------------------
   1   x'0103000000010000000500000000000000000000244000
       0000000000244000000000000000244000000000000003440
       0000000000003440000000000000344000000000000034
       4000000000000024400000000000002440000000000000
       2440'
```

Ejemplo 2:

El siguiente código en C muestra cómo utilizar el grupo de transformación ST_WellKnownBinary para insertar geometrías utilizando la función explícita ST_Geometry para la variable de lenguaje principal `wkb_buffer`, cuyo tipo es BLOB y que contiene la representación binaria convencional de una geometría que se debe insertar.

```
             EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS BLOB(1000) wkb_buffer;
```

```
 EXEC SQL END DECLARE SECTION;

// definir el grupo de transformación para todas las sentencias de SQL
// posteriores
 EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownBinary;

// inicializar variables de lenguaje principal
...
// insertar geometría con WKB en la columna de tipo ST_Geometry

 EXEC SQL
     INSERT
   INTO transforms_sample(id, geom)
    VALUES ( :id, :wkb_buffer );
```

Grupo de transformación ST_Shape

Utilice el grupo de transformación ST_Shape para transmitir los datos de y hacia DB2® utilizando la representación de forma de ESRI.

Cuando se vincula un valor del servidor de bases de datos con el cliente, se utiliza la misma función proporcionada por ST_AsShape() para convertir una geometría en su representación de forma. Cuando se transfiere la representación de forma de una geometría al servidor de bases de datos, se utiliza implícitamente la función ST_Geometry(BLOB) para realizar las conversiones a un valor de tipo ST_Geometry. El uso del grupo de transformación para vincular valores con DB2 hace que las geometrías se representen en el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Ejemplos:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido variará según la pantalla que se utilice.

Ejemplo 1:

El siguiente script de SQL muestra el uso del grupo de transformación ST_Shape para obtener una geometría en su representación forma, sin utilizar la función explícita ST_AsShape.

```
CREATE TABLE transforms_sample(
   id INTEGER,
   geom db2gse.ST_Geometry)

     INSERT
   INTO transforms_sample
   VALUES ( 1, db2gse.ST_Point(20.0, 30.0, 0) )

SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape

   SELECT id, geom
   FROM transforms_sample
   WHERE id = 1
```

Resultados:

```
ID     GEOM
----   ---------------------------------------------
   1    x'0100000000000000000034400000000000003E40'
```

Ejemplo 2:

El siguiente código en C muestra cómo utilizar el grupo de transformación ST_Shape para insertar geometrías utilizando la función explícita ST_Geometry para la variable de lenguaje principal `shape_buffer`, cuyo tipo es BLOB y que contiene la representación de forma de una geometría que se debe insertar.

```
             EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS BLOB(1000) shape_buffer;
 EXEC SQL END DECLARE SECTION;

// definir el grupo de transformación para todas las sentencias de SQL
// posteriores
 EXEC SQL
SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// inicializar variables de lenguaje principal
...

SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// insertar geometría con representación de forma en la columna de tipo ST_Geometry
 EXEC SQL
     INSERT
   INTO transforms_sample(id, geom)
    VALUES ( :id, :shape_buffer );
```

Grupo de transformación ST_GML

Utilice el grupo de transformación ST_GML para intercambiar datos con DB2® utilizando GML (Geography Markup Language).

Cuando se vincula un valor del servidor de bases de datos con el cliente, se utiliza la misma función proporcionada por ST_AsGML() para convertir una geometría en su representación GML. Cuando se transfiere la representación GML de una geometría al servidor de bases de datos, se utiliza implícitamente la función ST_Geometry(CLOB) para realizar las conversiones a un valor de tipo ST_Geometry. El uso del grupo de transformación para vincular valores con DB2 hace que las geometrías se representen en el sistema de referencia espacial cuyo identificador numérico es 0 (cero).

Ejemplos:

En los ejemplos siguientes, las líneas de resultados se han reformateado para mejorar la legibilidad. El espaciado de líneas en el resultado obtenido variará según la pantalla que se utilice.

Ejemplo 1:

El siguiente script de SQL muestra el uso del grupo de transformación ST_GML para obtener una geometría en su representación GML sin utilizar la función explícita ST_AsGML.

```
CREATE TABLE transforms_sample (
   id  INTEGER,
   geom  db2gse.ST_Geometry)

     INSERT
  INTO  transforms_sample
  VALUES ( 1, db2gse.ST_Geometry('multipoint z (10 10
           3, 20 20 4, 15 20 30)', 0) )
```

```
SET CURRENT DEFAULT TRANSFORM GROUP = ST_GML

   SELECT id, geom
FROM  transforms_sample
WHERE  id = 1
```

Resultados:

```
ID     GEOM
-----  ------------------------------------------------------------
    1 <gml:MultiPoint srsName=UNSPECIFIED><gml:PointMember>
      <gml:Point><gml:coord><gml:X>10</gml:X>
      <gml:Y>10</gml:Y><gml:Z>3</gml:Z>
      </gml:coord></gml:Point></gml:PointMember>
      <gml:PointMember><gml:Point><gml:coord>
      <gml:X>20</gml:X><gml:Y>20</gml:Y>
      <gml:Z>4</gml:Z></gml:coord></gml:Point>
      </gml:PointMember><gml:PointMember><gml:Point>
      <gml:coord><gml:X>15</gml:X><gml:Y>20
      </gml:Y><gml:Z>30</gml:Z></gml:coord>
      </gml:Point></gml:PointMember></gml:MultiPoint>
```

Ejemplo 2:

El siguiente código en C muestra cómo utilizar el grupo de transformación ST_GML para insertar geometrías sin utilizar la función explícita ST_Geometry para la variable de lenguaje principal `gml_buffer`, cuyo tipo es CLOB y que contiene la representación GML del punto (20 ,20) que se debe insertar.

```
             EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS CLOB(1000) gml_buffer;
 EXEC SQL END DECLARE SECTION;

// definir el grupo de transformación para todas las sentencias de SQL
// posteriores
 EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_GML;
 id = 100;
strcpy(gml_buffer.data, "<gml:point><gml:coord>"
   "<gml:X>20</gml:X> <gml:Y>20</gml:Y></gml:coord></gml:point>");

// inicializar variables de lenguaje principal
wkt_buffer.length = strlen(gml_buffer.data);

// insertar punto con WKT en la columna de tipo ST_Geometry
 EXEC SQL
     INSERT
   INTO transforms_sample(id, geom)
   VALUES ( :id, :gml_buffer );
```

Capítulo 25. Formatos de datos soportados

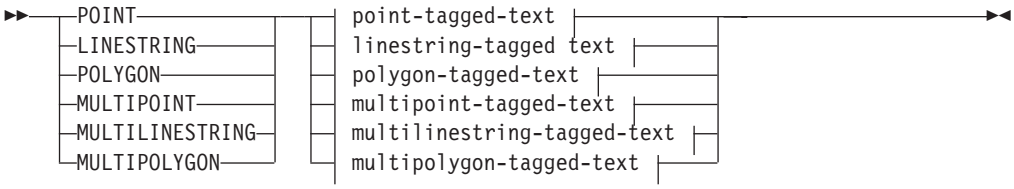
Este capítulo describe los formatos de datos espaciales aceptados que se pueden utilizar con DB2 Spatial Extender. Consulte el apartado "Funciones espaciales que convierten valores de geometría en formatos de intercambio de datos" en la página 307 para obtener información sobre las funciones que aceptan y crean estos formatos. Consulte el apartado "Acerca de la importación y la exportación de datos espaciales" en la página 89 para obtener información sobre la importación y exportación de archivos donde están contenidos estos formatos. Se describen estos cuatro formatos de datos espaciales:

- Representación de texto convencional (well-known text, WKT)
- Representación binaria convencional (well-known binary, WKB)
- Representación de forma
- Representación GML (Geography Markup Language)

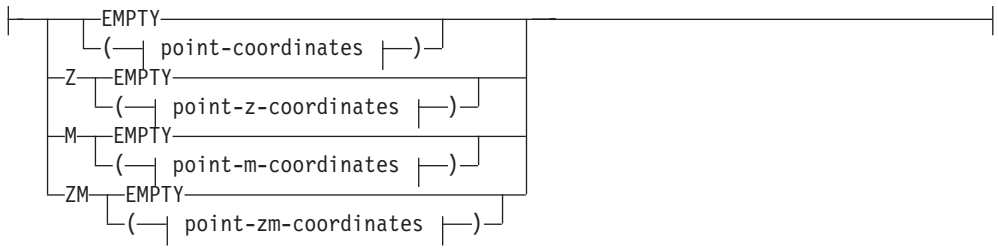
Representación de texto convencional (WKT)

La especificación ″Simple Features for SQL″ de OpenGIS Consortium define la representación de texto convencional para intercambiar datos de geometría en formato ASCII. Esta representación también está documentada en el estándar ″SQL/MM Part: 3 Spatial″ de ISO. Consulte el apartado ″Funciones espaciales que convierten geometrías entre formatos de intercambio de datos″ para obtener información sobre funciones que aceptan y producen datos en formato WKT.

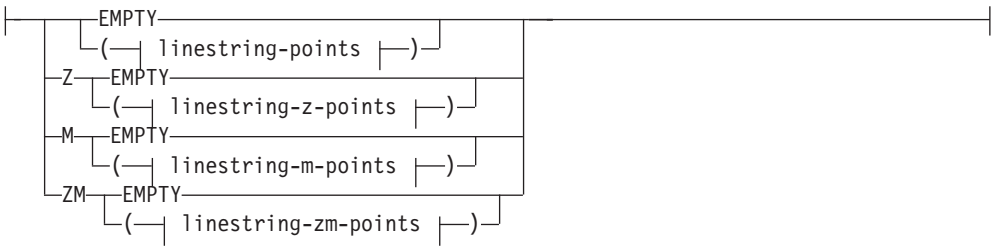
La representación de texto convencional de una geometría está definida de esta manera:



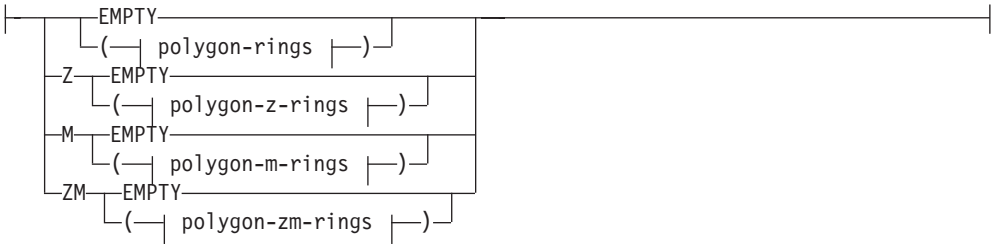
point-tagged-text:



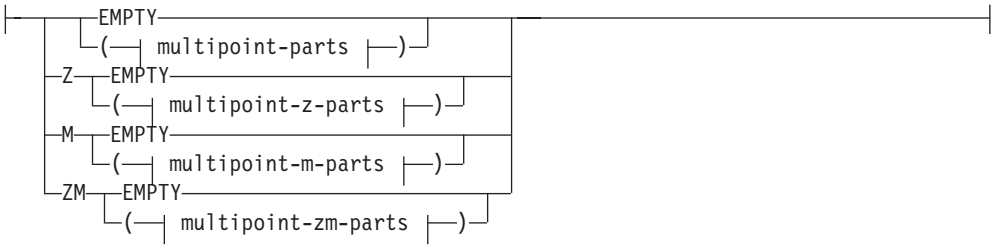
linestring-tagged-text:



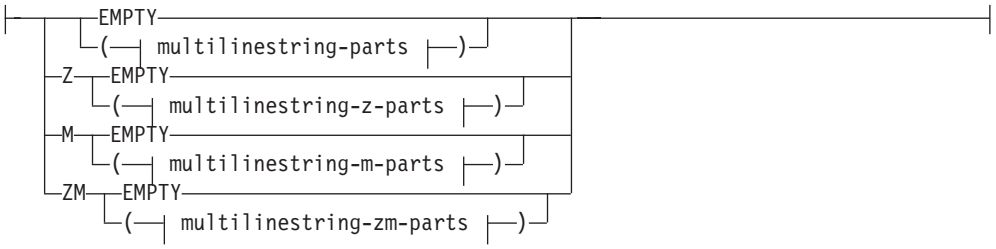
polygon-tagged-text:



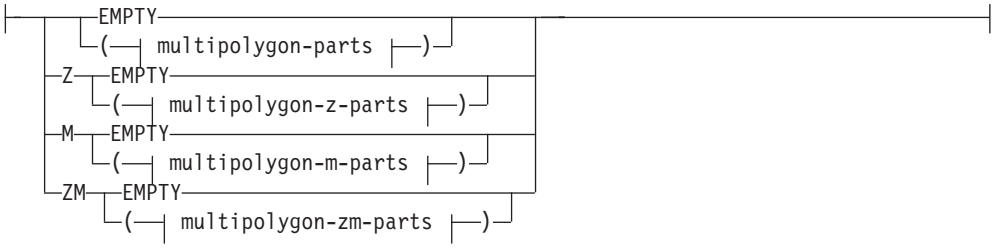
multipoint-tagged-text:



multilinestring-tagged-text:

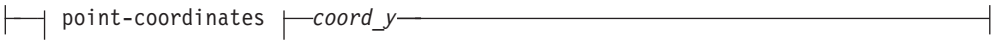


multipolygon-tagged-text:



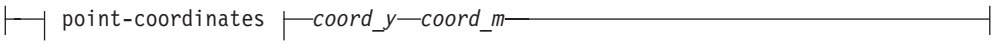
point-coordinates:

point-z-coordinates:

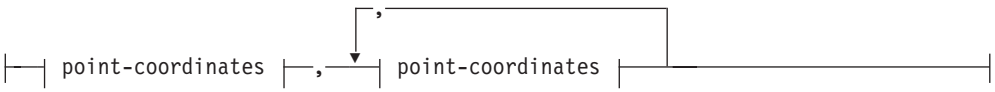


point-m-coordinates:

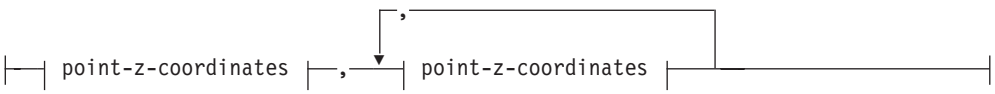
point-zm-coordinates:



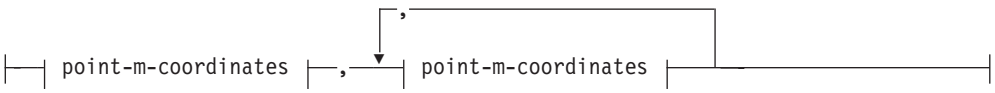
linestring-points:



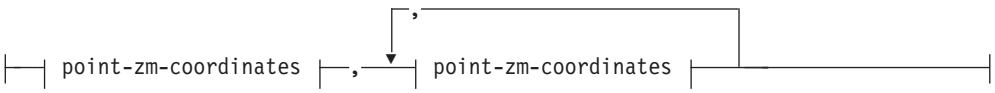
linestring-z-points:



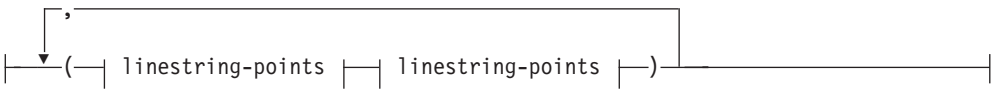
linestring-m-points:



linestring-zm-points:



polygon-rings:



polygon-z-rings:

polygon-m-rings:

polygon-zm-rings:

multipoint-parts:

multipoint-z-parts:

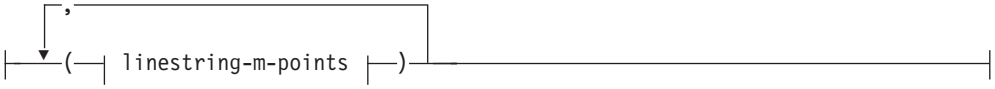
multipoint-m-parts:

multipoint-zm-parts:

multilinestring-parts:

multilinestring-z-parts:

multilinestring-m-parts:



multilinestring-zm-parts:

multipolygon-parts:

multipolygon-z-parts:

multipolygon-m-parts:

multipolygon-zm-parts:

Parámetros:
coord_x

Valor numérico (fijo, entero o de coma flotante) que representa la coordenada X de un punto.

coord_y

Valor numérico (fijo, entero o de coma flotante) que representa la coordenada Y de un punto.

coord_z

Valor numérico (fijo, entero o de coma flotante) que representa la coordenada Z de un punto.

coord_m

Valor numérico (fijo, entero o de coma flotante) que representa la coordenada M (medida) de un punto.

Si la geometría está vacía, se debe especificar la palabra clave EMPTY en lugar de la lista de coordenadas. La palabra clave EMPTY no debe estar incluida dentro de la lista de coordenadas.

La tabla siguiente proporciona algunos ejemplos de posibles representaciones de texto.

Tabla 57. Tipos de geometría y sus representaciones de texto

Tipo de geometría	Representación	Comentario
punto	POINT EMPTY	punto vacío
punto	POINT (10.05 10.28)	punto
punto	POINT Z(10.05 10.28 2.51)	punto con coordenada Z
punto	POINT M(10.05 10.28 4.72)	punto con coordenada M
punto	POINT ZM(10.05 10.28 2.51 4.72)	punto con coordenada Z y coordenada M
cadena lineal	LINESTRING EMPTY	cadena lineal vacía
polígono	POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))	polígono
multipunto	MULTIPOINT Z(10 10 2, 20 20 3)	multipunto con coordenadas Z
multilínea	MULTILINESTRING M((310 30 1, 40 30 20, 50 20 10)(10 10 0, 20 20 1))	multilínea con coordenadas M
multipolígono	MULTIPOLYGON ZM(((1 1 1 1, 1 2 3 4, 2 2 5 6, 2 1 7 8, 1 1 1 1)))	multipolígono con coordenadas Z y coordenadas M

Información relacionada:

- "Funciones espaciales que convierten valores de geometría en formatos de intercambio de datos" en la página 307

Representación binaria convencional (WKB)

Esta sección describe la representación binaria convencional de las geometrías.

La especificación "Simple Features for SQL" de OpenGIS Consortium define la representación binaria convencional. Esta representación también está definida en el estándar "SQL/MM Part: 3 Spatial" de ISO (International Organization for Standardization). Consulte la sección de consulta correspondiente, al final de este tema, para obtener información sobre funciones que aceptan y producen datos en formato WKB.

El componente básico de la representación binaria convencional es la corriente de bytes correspondiente a un punto, que consta de dos valores de tipo ″double″. Las corrientes de bytes correspondientes a otras geometrías se crean utilizando las corrientes de bytes de geometrías que ya están definidas.

El ejemplo siguiente muestra el componente básico de las representaciones binarias convencionales.

```
// Basic Type definitions
// byte : 1 byte
// uint32 : 32 bit unsigned integer  (4 bytes)
// double : double precision number (8 bytes)

// Building Blocks : 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=wkbPoint
  Point    point;
};
WKBLineString {
  byte     byteOrder;
  uint32   wkbType;     // 2=wkbLineString
  uint32   numPoints;
  Point    points[numPoints];
};

WKBPolygon    {
  byte               byteOrder;
  uint32            wkbType;     // 3=wkbPolygon
  uint32            numRings;
  LinearRing        rings[numRings];
};
WKBMultiPoint    {
  byte               byteOrder;
  uint32            wkbType;     // 4=wkbMultipoint
  uint32            num_wkbPoints;
  WKBPoint            WKBPoints[num_wkbPoints];
};
WKBMultiLineString    {
  byte              byteOrder;
  uint32            wkbType;     // 5=wkbMultiLineString
  uint32            num_wkbLineStrings;
  WKBLineString     WKBLineStrings[num_wkbLineStrings];
};
```

```
wkbMultiPolygon {
  byte              byteOrder;
  uint32            wkbType;     // 6=wkbMultiPolygon
  uint32            num_wkbPolygons;
  WKBPolygon        wkbPolygons[num_wkbPolygons];
};

WKBGeometry  {
  union {
    WKBPoint                point;
    WKBLineString           linestring;
    WKBPolygon              polygon;
    WKBMultiPoint           mpoint;
    WKBMultiLineString      mlinestring;
    WKBMultiPolygon         mpolygon;
  }
};
```

La figura siguiente es un ejemplo de geometría en representación binaria convencional que hace uso de la codificación NDR.

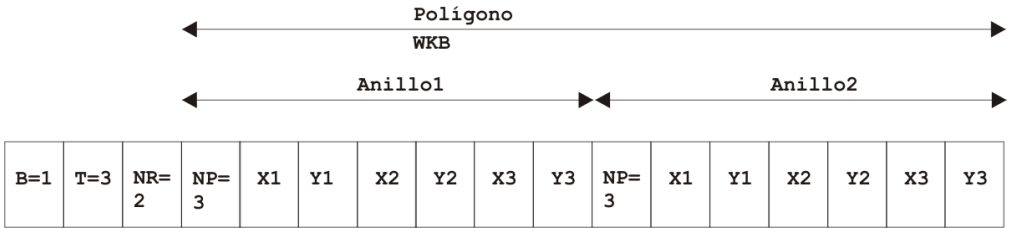


Figura 58. Representación de una geometría en formato NDR. (B=1) de tipo polígono (T=3) con 2 lineales (NR=2), donde cada anillo tiene 3 puntos (NP=3).

Información relacionada:

- "Funciones espaciales que convierten valores de geometría en formatos de intercambio de datos" en la página 307

Representación de forma

La representación de forma es un estándar de uso habitual definido por ESRI. Para obtener una descripción completa de la representación de forma, consulte el sitio Web de ESRI, situado en http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

El apartado "Funciones espaciales que convierten geometrías entre formatos de intercambio de datos", en la sección de enlaces afines mostrada más abajo, describe las funciones espaciales que aceptan y producen datos en el formato de forma.

Información relacionada:

- "Funciones espaciales que convierten valores de geometría en formatos de intercambio de datos" en la página 307

Representación GML (Geography Markup Language)

DB2 Spatial Extender tiene varias funciones que crean geometrías a partir de representaciones en formato GML (Geography Markup Language).

Consulte más abajo el enlace asociado ″Funciones espaciales que convierten geometrías entre formatos de intercambio de datos″ para obtener una descripción detallada de las funciones proporcionadas por DB2 Spatial Extender que convierten valores de geometría a partir de una representación GML.

GML (Geography Markup Language) es una codificación XML para información geográfica que está definida por la especificación ″Geography Markup Language V2″ de OpenGIS Consortium. Puede encontrar esta especificación de OpenGIS Consortium en este sitio Web:
http://www.opengis.org/techno/implementation.htm.

Información relacionada:

- “Funciones espaciales que convierten valores de geometría en formatos de intercambio de datos” en la página 307

Capítulo 26. Sistemas de coordenadas soportados

Este capítulo proporciona información de consulta sobre los valores de coordenadas utilizados para interpretar los datos espaciales. Se tratan los temas siguientes:

- Visión general de los sistemas de coordenadas
- Unidades lineales soportadas
- Unidades angulares soportadas
- Esferoides soportados
- Datums geodésicos soportados
- Meridianos de origen soportados
- Proyecciones cartográficas soportadas

Sistemas de coordenadas soportados

Este tema describe la sintaxis de los sistemas de coordenadas y lista los valores de sistemas de coordenadas que están soportados por DB2 Spatial Extender.

Sintaxis de los sistemas de coordenadas

La representación de texto convencional de los sistemas de referencia espacial proporciona una representación textual estándar para la información sobre sistemas de coordenadas. Las definiciones de la representación de texto convencional están definidas en la especificación ″Simple Features for SQL″ de OGC y el estándar ISO SQL/MM Part 3: Spatial.

Un sistema de coordenadas es un sistema de coordenadas geográficas (latitud-longitud), un sistema de coordenadas proyectadas (X,Y) o un sistema de coordenadas geocéntricas (X,Y,Z). El sistema de coordenadas consta de varios objetos. Cada objeto tiene asignada una palabra clave escrita en mayúsculas (por ejemplo, DATUM o UNIT) seguida entre paréntesis o corchetes por los parámetros que definen el objeto, separados por comas. Algunos objetos están formados por otros objetos, por lo que el resultado es una estructura anidada.

Nota: En la práctica se pueden utilizar paréntesis estándar ( ) en lugar de corchetes [ ] y debe ser posible leer ambos tipos de notación.

La definición EBNF (Extended Backus Naur Form) para la representación en forma de cadena de caracteres de un sistema de coordenadas utilizando corchetes es la siguiente (consulte la nota anterior sobre el uso de corchetes):

```
<sistema de coordenadas> = <sc proyectadas> |
<sc geográficas> | <sc geocéntricas cs>
<sc proyectadas> = PROJCS["<nombre>",
<sc geográficas>, <proyección>, {<parámetro>,}*
<unidad lineal>]
<proyección> = PROJECTION["<nombre>"]
<parámetro> = PARAMETER["<nombre>",
<valor>]

<valor> = <número>
```

El tipo de sistema de coordenadas está identificado por la palabra clave utilizada:

PROJCS
Esta palabra clave identifica un sistema de coordenadas de un conjunto de datos cuando los datos están en coordenadas proyectadas

GEOGCS
Esta palabra clave identifica un sistema de coordenadas de un conjunto de datos cuando los datos están en coordenadas geográficas

GEOCCS
Esta palabra clave identifica un sistema de coordenadas de un conjunto de datos cuando los datos están en coordenadas geocéntricas

La palabra clave PROJCS está seguida por todos los elementos que definen el sistema de coordenadas proyectadas. El primer elemento de cualquier objeto es siempre el nombre. El nombre del sistema de coordenadas proyectadas va seguido por varios objetos: el sistema de coordenadas geográficas, la proyección cartográfica, uno o varios parámetros y la unidad lineal de medida. Todos los sistemas de coordenadas proyectadas están basados en un sistema de coordenadas geográficas, por lo que esta sección describe primero los elementos específicos de un sistema de coordenadas proyectadas. Por ejemplo, la zona 10N de UTM sobre el datum NAD83 está definida así:

```
PROJCS["NAD_1983_UTM_Zone_10N",
<sc geográficas>,
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 del sistema de coordenadas geográficas secuencialmente: el datum, el meridiano de origen y la unidad angular de medida.

```
<sc geográficas> = GEOGCS["<nombre>", <datum>, <meridiano de origen>,
<unidad angular>]
<datum> = DATUM["<nombre>", <esferoide>]
<esferoide> = SPHEROID["<nombre>", <eje semimayor>, <aplanamiento inverso>]
<eje semimayor> = <número>
<aplanamiento inverso> = <número>
<meridiano de origen> = PRIMEM["<nombre>", <longitud>]
<longitud> = <número>
```

El eje semimayor se mide en metros y debe ser mayor que cero.

Esta sería la definición del sistema de coordenadas geográficas para la zona 10 de UTM 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 una unidad angular o lineal de medidas:

```
<unidad angular> = <unidad>
<unidad lineal> = <unidad>
<unidad> = UNIT["<nombre>", <factor de conversión>]
<factor de conversión> = <número>
```

El factor de conversión especifica el número de metros (para una unidad lineal) o el número de radianes (para una unidad angular) por cada unidad y debe ser mayor que cero.

Por tanto, la representación completa en forma de cadena de caracteres de la zona 10N de UTM sería 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 similar a un sistema de coordenadas geográficas:

```
<sc geocéntricas> = GEOCCS["<nombre>", <datum>, <meridiano de origen>,
<unidad lineal>]
```

Unidades lineales soportadas

Tabla 58. Unidades lineales soportadas

Unidad	Factor de conversión
Metro	1,0
Pie (Internacional)	0,3048
Pie americano	12/39,37
Pie americano modificado	12,0004584/39,37
Pie de Clarke	12/39,370432
Pie indio	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 (India)	36/39,370141
Yarda (Sears)	36/39,370147
Braza	1,8288
Milla náutica	1852,0

Unidades angulares soportadas

Tabla 59. Unidades angulares soportadas

Unidad	Rango válido para la latitud	Rango válido para la longitud	Factor de conversión
Radián	radianes –pi/2 y pi/2 (inclusive)	radianes –pi y pi (inclusive)	1,0
Grado decimal	–90 y 90 grados (inclusive)	–180 y 180 grados (inclusive)	pi/180

Tabla 59. Unidades angulares soportadas (continuación)

Unidad	Rango válido para la latitud	Rango válido para la longitud	Factor de conversión
Minuto decimal	–5400 y 5400 minutos (inclusive)	–10800 y 10800 minutos (inclusive)	(pi/180)/60
Segundo decimal	–324000 y 324000 segundos (inclusive)	–648000 y 648000 segundos (inclusive)	(pi/180)*3600
Gon	–100 y 100 gradianes (inclusive)	–200 y 200 gradianes (inclusive)	pi/200
Grad	–100 y 100 gradianes (inclusive)	–200 y 200 gradianes (inclusive)	pi/200

Esferoides soportados

Tabla 60. Esferoides soportados

Nombre	Eje semimayor	Aplanamiento inverso
Airy 1830	6377563,396	299,3249646
Airy Modified 1849	6377340,189	299,3249646
Average Terrestrial System 1977	6378135,0	298,257
Australian National Spheroid	6378160,0	298,25
Bessel 1841	6377397,155	299,1528128
Bessel Modified	6377492,018	299,1528128
Bessel Namibia	6377483,865	299,1528128
Clarke 1858	6378293.639	294.260676369
Clarke 1866	6378206,4	294,9786982
Clarke 1866 (Michigan)	6378450,047	294,978684677
Clarke 1880	6378249,138	293,466307656
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 1922)	6378249,2	293,46598
Everest (1830 Definition)	6377299,36	300,8017
Everest 1830 Modified	6377304,063	300,8017
Everest Adjustment 1937	6377276,345	300,8017
Everest 1830 (1962 Definition)	6377301,243	300,8017255
Everest 1830 (1967 Definition)	6377298,556	300,8017
Everest 1830 (1975 Definition)	6377299,151	300,8017255
Everest 1969 Modified	6377295,664	300,8017
Fischer 1960	6378166,0	298,3
Fischer 1968	6378150,0	298,3
Modified Fischer	6378155,0	298,3
GEM 10C	6378137,0	298,257222101

Tabla 60. Esferoides soportados (continuación)

Nombre	Eje semimayor	Aplanamiento inverso
GRS 1967	6378160,0	298,247167427
GRS 1967 Truncated	6378160,0	298,25
GRS 1980	6378137,0	298,257222101
Helmert 1906	6378200,0	298,3
Hough 1960	6378270,0	297,0
Indonesian National Spheroid	6378160,0	298,247
International 1924	6378388,0	297,0
International 1967	6378160,0	298,25
Krassowsky 1940	6378245,0	298,3
NWL 9D	6378145,0	298,25
NWL 10D	6378135,0	298,26
OSU 86F	6378136,2	298,25722
OSU 91A	6378136,3	298,25722
Plessis 1817	6376523,0	308,64
Sphere	6371000,0	0,0
Sphere (ArcInfo)	6370997,0	0,0
Struve 1860	6378298,3	294,73
Walbeck	6376896,0	302,78
War Office	6378300,0	296,0
WGS 1966	6378145,0	298,25
WGS 1972	6378135,0	298,26
WGS 1984	6378137,0	298,257223563

Datums geodésicos soportados

Tabla 61. Datums geodésicos soportados

Nombre	Datum geodésico
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

Tabla 61. Datums geodésicos soportados (continuación)

Nombre	Datum geodésico
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
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
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
Bogotá	Nord de Guerre
Bukit Rimpah	NGO 1948
Camacupa	Nord Sahara 1959
Campo Inchauspe	NSWC 9Z-2
Cape	Nouvelle Triangulation Francaise
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

Tabla 61. Datums geodésicos soportados (continuación)

Nombre	Datum geodésico
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
Leigon	Yoff
Liberia 1964	Zanderij

Meridianos de origen soportados

Tabla 62. Meridianos de origen soportados

Ubicación	Coordenadas
Greenwich	0° 0' 0"
Berna	7° 26' 22,5" E
Bogotá	74° 4' 51,3" O
Bruselas	4° 22' 4,71" E
Ferro	17° 40' 0" O
Yakarta	106° 48' 27,79" E
Lisboa	9° 7' 54,862" O
Madrid	3° 41' 16,58" O

Tabla 62. Meridianos de origen soportados (continuación)

Ubicación	Coordenadas
París	2° 20' 14,025" E
Roma	12° 27' 8,4" E
Estocolmo	18° 3' 29" E

Proyecciones cartográficas soportadas

Tabla 63. Proyecciones cilíndricas

Proyecciones cilíndricas	Proyecciones seudocilíndricas
Behrmann	Parabólica de Craster
Cassini	Eckert I
Cilíndrica de igual área	Eckert II
Equirrectangular	Eckert III
Estereográfica de Gall	Eckert IV
Gauss-Kruger	Eckert V
Mercator	Eckert VI
Cilíndrica de Miller	Cuártica polar plana de McBryde-Thomas
Oblicua	Mercator (Hotine) Mollweide
Plate-Carée	Robinson
Times	Sinusoidal (Sansom-Flamsteed)
Transversal de Mercator	Winkel I

Tabla 64. Proyecciones cónicas

Nombre	Proyección cónica
Cónica de igual área de Albers	Trimétrica de Chamberlin
Cónica conforme oblicua bipolar	Equidistante de dos puntos
Bonne	Hammer-Aitoff de igual área
Cónica equidistante	Van der Grinten I
Cónica conforme de Lambert	Varios
Policónica	Alaska serie E
Cónica simple	Alaska Grid (Estereográdica modificada por Snyder)

Tabla 65. Parámetros de la proyección cartográfica

Parámetro	Descripción
central_meridian	Línea de longitud elegida como origen de las coordenadas X.
scale_factor	Es un factor de escala que se utiliza generalmente para disminuir el grado de distorsión en una proyección cartográfica.
standard_parallel_1	Línea de latitud que básicamente carece de distorsión. También se utiliza para la ″latitud de escala verdadera″.

Tabla 65. Parámetros de la proyección cartográfica (continuación)

Parámetro	Descripción
standard_parallel_2	Línea de longitud que básicamente carece de distorsión.
longitude_of_center	Longitud que define el punto central de la proyección cartográfica.
latitude_of_center	Latitud que define el punto central de la proyección cartográfica.
longitude_of_origin	Longitud elegida como origen de las coordenadas X.
latitude_of_origin	Latitud elegida como origen de las coordenadas Y.
false_easting	Valor que se agrega a las coordenadas X para que todos sus valores sean positivos.
false_northing	Valor que se agrega a las coordenadas Y para que todos sus valores sean positivos.
azimuth	Ángulo hacia el este del norte que define la línea central de una proyección oblicua.
longitude_of_point_1	Longitud del primer punto necesario para una proyección cartográfica.
latitude_of_point_1	Latitud del primer punto necesario para una proyección cartográfica.
longitude_of_point_2	Longitud del segundo punto necesario para una proyección cartográfica.
latitude_of_point_2	Latitud del segundo punto necesario para una proyección cartográfica.
longitude_of_point_3	Longitud del tercer punto necesario para una proyección cartográfica.
latitude_of_point_3	Latitud del tercer punto necesario para una proyección cartográfica.
landsat_number	Número de un satélite Landsat.
path_number	Número de la ruta orbital de un determinado satélite.
perspective_point_height	Altura sobre la superficie terrestre del punto de perspectiva de la proyección cartográfica.
fipszone	Número de zona del sistema de coordenadas de Plano de estado.
zone	Número de zona de UTM.

Apéndice A. Procedimientos almacenados obsoletos

Este tema describe los procedimientos almacenados que han quedado obsoletos.

Nota: Recomendación: escriba todas las nuevas aplicaciones utilizando los procedimientos almacenados definidos en DB2 Spatial Extender Versión 8 y actualice las aplicaciones actuales para utilizar los procedimientos almacenados definidos en la Versión 8.

Los procedimientos almacenados obsoletos realizaban las tareas resumidas en la tabla siguiente.

Tabla 66. Procedimientos almacenados obsoletos

Nombre del procedimiento almacenado	Tarea del procedimiento almacenado
db2gse.gse_enable_autogc	Habilita un geocodificador para mantener automáticamente sincronizadas las columnas espaciales con sus correspondientes columnas de atributos
db2gse.gse_enable_db	Habilita una base de datos para dar soporte a las operaciones espaciales
db2gse.gse_enable_idx	Crea un índice para una columna espacial
db2gse.gse_enable_sref	Crea un sistema de referencia espacial
db2gse.gse_export_shape	Exporta una capa y su tabla asociada a un archivo de formas
db2gse.gse_disable_autogc	Inhabilita un geocodificador para que no pueda mantener automáticamente sincronizadas las columnas espaciales con sus correspondientes columnas de atributos
db2gse.gse_disable_db	Inhabilita el soporte para operaciones espaciales en una base de datos
db2gse.gse_disable_sref	Descarta un sistema de referencia espacial
db2gse.gse_import_shape	Importa una capa y su tabla asociada desde un archivo de transferencia ESRI_SDE
db2gse.gse_register_gc	Registra un geocodificador distinto del geocodificador por omisión
db2gse.gse_register_layer	Registra una columna espacial como capa
db2gse.gse_run_gc	Ejecuta un geocodificador en la modalidad de proceso por lotes
db2gse.gse_unregist_gc	Desregistra un geocodificador distinto del geocodificador por omisión
db2gse.gse_unregist_layer	Desregistra una capa

db2gse.gse_enable_autogc

Utilice este procedimiento almacenado para:

- Crear desencadenantes que mantienen sincronizada una columna espacial con su correspondiente columna o columnas de atributos. Cada vez que se insertan o actualizan valores en la columna o columnas de atributos, un desencadenante

llama a un geocodificador registrado para geocodificar los valores insertados o actualizados y colocar los datos resultantes en la columna espacial.
- Reactivar los desencadenantes después de su inhabilitación temporal.
- Establecer qué función se debe utilizar para geocodificar los valores insertados y actualizados.

Autorización:

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener una de las autorizaciones o privilegios siguientes:
- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla donde están definidos los desencadenantes que ha creado este procedimiento almacenado.
- Privilegio CONTROL sobre esta tabla.
- Privilegios ALTER, SELECT y UPDATE sobre esta tabla.

Parámetros:

Tabla 67. Parámetros de entrada del procedimiento almacenado db2gse.gse_enable_autogc.

Nombre	Tipo de datos	Descripción
operMode	SMALLINT	Valor que indica si los desencadenantes que inician la geocodificación se deben crear por primera vez o se deben reactivar después de su inhabilitación temporal. Este parámetro no puede tener un valor nulo. **Comentario:** Para crear los desencadenantes, utilice la macro GSE_AUTOGC_CREATE. Para reactivarlos, utilice la macro GSE_AUTOGC_RECREATE. Para conocer qué valores se deben utilizar con estas macros, consulte el archivo db2gse.h. En AIX, este archivo está almacenado en el directorio $DB2INSTANCE/sqllib/include/. En Windows NT, el archivo reside en el directorio %DB2PATH%\include\. Si el parámetro operMode tiene el valor GSE_AUTOGC_CREATE, debe asignar un identificador de un geocodificador registrado al parámetro gcId.
layerSchema	VARCHAR(30)	Nombre del esquema al que pertenece la tabla especificada en el parámetro layerTable. Este parámetro puede tener un valor nulo. Si no proporciona un valor para el parámetro layerSchema, se establecerá por omisión en el ID de usuario utilizado para invocar al procedimiento almacenado db2gse.gse_enable_autogc.
layerTable	VARCHAR(128)	Nombre de la tabla sobre la que deben actuar los desencadenantes creados o reactivados por este procedimiento almacenado. Este parámetro no puede tener un valor nulo.

Tabla 67. Parámetros de entrada del procedimiento almacenado db2gse.gse_enable_autogc. (continuación)

Nombre	Tipo de datos	Descripción
layerColumn	VARCHAR(128)	Nombre de la columna espacial que debe ser mantenida por los desencadenantes creados o reactivados por este procedimiento almacenado. Este parámetro no puede tener un valor nulo. El parámetro layerColumn debe hacer referencia a una columna que se ha registrado como capa de tabla.
gcId	INTEGER	Identificador del geocodificador que será invocado por los desencadenantes de inserción y actualización que este procedimiento almacenado crea o reactiva. Este parámetro no puede tener un valor nulo si el parámetro operMode tiene el valor GSE_AUTOGC_CREATE. Puede tener un valor nulo si operMode es igual a GSE_AUTOGC_RECREATE.
precisionLevel	INTEGER	Grado de coincidencia que debe existir entre los datos de origen y los correspondientes datos de referencia para que el geocodificador procese los datos de origen satisfactoriamente. Este parámetro no puede tener un valor nulo si el parámetro operMode tiene el valor GSE_AUTOGC_CREATE. Puede tener un valor nulo si operMode es igual a GSE_AUTOGC_RECREATE. El nivel de precisión puede tener un valor comprendido entre el 1 y el 100 por ciento.
vendorSpecific	VARCHAR(256)	Información técnica proporcionada por el proveedor; por ejemplo, el nombre y vía de acceso de un archivo que el proveedor utiliza para definir parámetros. Este parámetro no puede tener un valor nulo si el parámetro operMode tiene el valor GSE_AUTOGC_CREATE. Puede tener un valor nulo si operMode es igual a GSE_AUTOGC_RECREATE.

Resultados:

Tabla 68. Parámetros de salida del procedimiento almacenado db2gse.gse_enable_autogc.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor.

db2gse.gse_enable_db

Utilice este procedimiento almacenado para proporcionar a la base de datos los recursos que necesita para almacenar datos espaciales y para dar soporte a operaciones espaciales. Estos recursos incluyen tipos de datos espaciales, un tipo de índice espacial, tablas y vistas de catálogo, funciones proporcionadas y otros procedimientos almacenados. La biblioteca externa y el nombre de función para este procedimiento almacenado es db2gse.gse_enable_db.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos que se está habilitando.

Resultados:

Tabla 69. Parámetros de salida del procedimiento almacenado db2gse.gse_enable_db.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor de DB2 Spatial Extender.

db2gse.gse_enable_idx

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

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla para la que se debe utilizar el índice habilitado.
- Privilegio CONTROL o INDEX sobre esta tabla.

Parámetros:

Tabla 70. Parámetros de entrada del 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 puede tener un valor nulo. Debe proporcionar un valor para este parámetro. El parámetro puede ser un valor nulo.
layerTable	VARCHAR(128)	Nombre de la tabla para la que se debe definir el índice que se está creando. Este parámetro no puede tener un valor nulo.

Tabla 70. Parámetros de entrada del procedimiento almacenado db2gse.gse_enable_idx. (continuación)

Nombre	Tipo de datos	Descripción
layerColumn	VARCHAR(128)	Nombre de la columna habilitada para operaciones espaciales en la que se debe buscar mediante la ayuda del índice que se está creando. Este parámetro no puede tener un valor nulo.
indexName	VARCHAR(128)	Nombre del índice que se debe crear. Este parámetro no puede tener un valor nulo. No especifique un nombre de esquema. DB2 Spatial Extender asigna automáticamente el índice al esquema especificado por el parámetro layerSchema.
gridSize1	DOUBLE	Número que indica cuál debe ser la granularidad de la retícula de índice más fina. Este parámetro no puede tener un valor nulo.
gridSize2	DOUBLE	Número que indica: (1) que no debe existir una segunda retícula para este índice o (2) cuál debe ser la granularidad de la segunda retícula. Este parámetro puede tener un valor nulo. Si no debe existir una segunda retícula, especifique un 0. Si desea una segunda retícula, debe tener una granularidad menor que la especificada por gridSize1.
gridSize3	DOUBLE	Número que indica: (1) que no debe existir una tercera retícula para este índice o (2) cuál debe ser la granularidad de la tercera retícula. Este parámetro puede tener un valor nulo. Si no debe existir una tercera retícula, especifique un 0. Si desea una tercera retícula, debe tener una granularidad menor que la especificada por gridSize2.

Resultados:

Tabla 71. Parámetros de salida del procedimiento almacenado db2gse.gse_enable_idx.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor de DB2 Spatial Extender.

db2gse.gse_enable_sref

Utilice este procedimiento almacenado para especificar cómo los números decimales negativos de un determinado sistema de coordenadas se deben convertir en números enteros positivos, para que puedan ser guardados por DB2 Spatial Extender. Estas especificaciones constituyen un *sistema de referencia espacial*. Cuando

se procesa este procedimiento almacenado, la información acerca del sistema de referencia espacial se añade a la vista de catálogo DB2GSE.SPATIAL_REF_SYS.

Autorización:

No se necesita ninguna.

Parámetros:

Tabla 72. Parámetros de entrada del procedimiento almacenado db2gse.gse_enable_sref.

Nombre	Tipo de datos	Descripción
srId	INTEGER	Identificador numérico del sistema de referencia espacial. Este identificador debe ser exclusivo dentro de su base de datos habilitada para operaciones espaciales. Este parámetro no puede tener un valor nulo.
srName	VARCHAR(64)	Descripción abreviada del sistema de referencia espacial. Esta descripción debe ser exclusiva dentro de su base de datos habilitada para operaciones espaciales. Este parámetro no puede tener un valor nulo.
falsex	DOUBLE	Número que se resta de un valor de coordenada X negativo y da como resultado un número positivo o el valor 0. Este parámetro no puede tener un valor nulo.
falsey	DOUBLE	Número que se resta de un valor de coordenada Y negativo y da como resultado un número positivo o el valor 0. Este parámetro no puede tener un valor nulo.
xyunits	DOUBLE	Número que se multiplica por un valor decimal de coordenada X o Y y da como resultado un valor entero que se puede guardar como dato de 32 bits. Este parámetro no puede tener un valor nulo.
falsez	DOUBLE	Número que se resta de un valor de coordenada Z negativo y da como resultado un número positivo o el valor 0. Este parámetro no puede tener un valor nulo.
zunits	DOUBLE	Número que se multiplica por un valor decimal de coordenada Z y da como resultado un valor entero que se puede guardar como dato de 32 bits. Este parámetro no puede tener un valor nulo.
falsem	DOUBLE	Número que se resta de un valor de medida negativo y da como resultado un número positivo o el valor 0. Este parámetro no puede tener un valor nulo.

Tabla 72. Parámetros de entrada del procedimiento almacenado db2gse.gse_enable_sref. (continuación)

Nombre	Tipo de datos	Descripción
munits	DOUBLE	Número que se multiplica por un valor decimal de medida y da como resultado un valor entero que se puede guardar como dato de 32 bits. Este parámetro no puede tener un valor nulo.
scId	INTEGER	Identificador numérico del sistema de coordenadas del cual se obtiene el sistema de referencia espacial. Para determinar el identificador numérico de un sistema de coordenadas, consulte la vista de catálogo DB2GSE.COORD_REF_SYS. Este parámetro no puede tener un valor nulo.

Resultados:

Tabla 73. Parámetros de salida del procedimiento almacenado db2gse.gse_enable_sref.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea 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 formas, o para crear un nuevo archivo de formas y exportar una capa y su tabla a este nuevo archivo.

Autorización:

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener el privilegio SELECT sobre la tabla que se debe exportar.

Parámetros:

Tabla 74. Parámetros de entrada del 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 puede tener un valor nulo. Si no proporciona un valor para el parámetro layerSchema, se establecerá por omisión en el ID de usuario utilizado para invocar al procedimiento almacenado db2gse.gse_export_shape.
layerTable	VARCHAR(128)	Nombre de la tabla que está exportando. Este parámetro no puede tener un valor nulo.

Tabla 74. Parámetros de entrada del procedimiento almacenado db2gse.gse_export_shape. (continuación)

Nombre	Tipo de datos	Descripción
layerColumn	VARCHAR(30)	Nombre de la columna que se ha registrado como la capa que está exportando. Este parámetro no puede tener un valor nulo.
fileName	VARCHAR(128)	Nombre del archivo de formas al que se debe exportar la capa especificada. Este parámetro no puede tener un valor nulo.
whereClause	VARCHAR(1024)	Es el cuerpo de la cláusula whereClause. Define una restricción sobre el conjunto de filas que se deben exportar. La cláusula puede especificar cualquier columna de atributos de la tabla que está exportando. La palabra clave WHERE no es necesaria en esta cláusula. Este parámetro puede tener un valor nulo.

Resultados:

Tabla 75. Parámetros de salida del procedimiento almacenado db2gse.gse_export_shape.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor de DB2 Spatial Extender.

Restricción:

Puede exportar una sola capa cada vez.

db2gse.gse_disable_autogc

Utilice este procedimiento almacenado para descartar o inhabilitar temporalmente desencadenantes que mantienen sincronizada una columna espacial con su columna o columnas de atributos asociada. Por ejemplo, es recomendable inhabilitar los desencadenantes mientras geocodifica valores en la columna o columnas de atributos en la modalidad de proceso por lotes.

Autorización:

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla en la que están definidos los desencadenantes que se van a descartar o inhabilitar temporalmente.
- Privilegio CONTROL sobre esta tabla.
- Privilegios ALTER y UPDATE sobre esta tabla.

Nota: Para los privilegios CONTROL y ALTER, el usuario debe tener autorización DROPIN sobre el esquema DB2GSE.

Parámetros:

Tabla 76. Parámetros de entrada del procedimiento almacenado db2gse.gse_disable_autogc.

Nombre	Tipo de datos	Descripción
operMode	SMALLINT	Indica si los desencadenantes se deben descartar o inhabilitar temporalmente. Los desencadenantes descartados no tienen ningún efecto sobre las sentencias de SQL. Los desencadenantes inhabilitados temporalmente se pueden volver a crear sin tener que especificar de nuevo parámetros definidos previamente. Este parámetro no puede tener un valor nulo. Para descartar desencadenantes, utilice la macro GSE_AUTOGC_DROP. Para inhabilitarlos temporalmente, utilice la macro GSE_AUTOGC_INVALIDATE. Para conocer qué valores se deben utilizar con estas macros, consulte el archivo db2gse.h. En AIX, este archivo está almacenado en el directorio $DB2INSTANCE/sqllib/include/. En Windows NT, el archivo reside en el directorio %DB2PATH%\include\.
layerSchema	VARCHAR(30)	Nombre del esquema al que pertenece la tabla o vista especificada en el parámetro layerTable. Este parámetro puede tener un valor nulo. Si no proporciona un valor para el parámetro layerSchema, se establecerá por omisión en el ID de usuario utilizado para invocar al procedimiento almacenado db2gse.gse_disable_autogc.
layerTable	VARCHAR(128)	Nombre de la tabla en la que están definidos los desencadenantes que desea descartar o inhabilitar temporalmente. Este parámetro no puede tener un valor nulo.
layerColumn	VARCHAR(128)	Nombre de la columna habilitada para operaciones espaciales que se mantiene sincronizada mediante los desencadenantes que desea descartar o inhabilitar temporalmente. Este parámetro no puede tener un valor nulo.

Resultados:

Tabla 77. Parámetros de salida del procedimiento almacenado db2gse.gse_disable_autogc.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.

Tabla 77. Parámetros de salida del procedimiento almacenado db2gse.gse_disable_autogc. (continuación)

Nombre	Tipo de datos	Descripción
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea 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.

La finalidad de este procedimiento almacenado es ayudar al usuario a corregir problemas que surgen después de habilitar la base de datos para operaciones espaciales, pero *antes* de añadir columnas de tabla o datos espaciales a ella. Por ejemplo, si, después de habilitar una base de datos para operaciones espaciales, decide utilizar DB2 Spatial Extender para otra base de datos en lugar de aquélla. Si no ha definido ninguna columna espacial ni importado datos espaciales, puede invocar a este procedimiento almacenado para eliminar todos los recursos espaciales de la primera base de datos.

Autorización:

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos de la que se van a eliminar los recursos de DB2 Spatial Extender.

Resultados:

Tabla 78. Parámetros de salida del procedimiento almacenado db2gse.gse_disable_db.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor de DB2 Spatial Extender.

db2gse.gse_disable_sref

Utilice este procedimiento almacenado para descartar un sistema de referencia espacial. Cuando se procesa este procedimiento almacenado, la información acerca del sistema de referencia espacial se elimina de la vista de catálogo DB2GSE.SPATIAL_REF_SYS.

Requisitos previos:

Para descartar un sistema de referencia espacial, debe desregistrar las capas que hagan uso de él. Si tales capas permanecen desregistradas, se rechazará la petición para descartar el sistema de referencia espacial.

Autorización:

No se necesita ninguna.

Proceso:

Tabla 79. Parámetro de entrada del procedimiento almacenado db2gse.gse_disable_sref.

Nombre	Tipo de datos	Descripción
srId	INTEGER	Identificador numérico del sistema de referencia espacial que se debe descartar. Este parámetro no puede tener un valor nulo.

Resultados:

Tabla 80. Parámetros de salida del procedimiento almacenado db2gse.gse_disable_sref.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor de DB2 Spatial Extender.

db2gse.gse_import_shape

Utilice este procedimiento almacenado para importar un archivo de formas ESRI a una base de datos que está habilitada para operaciones espaciales. El procedimiento almacenado puede funcionar de cualquiera de estas dos maneras:

- Si el archivo de formas es para una tabla existente que tiene una columna de capa registrada, DB2 Spatial Extender carga la tabla con los datos del archivo.
- Si el archivo de formas es para una tabla que no existe, DB2 Spatial Extender crea una tabla que tiene una columna espacial, registra esta columna como capa, y carga la capa y las otras columnas de la tabla con los datos del archivo.

Cuando importa un conjunto de representaciones de formas ESRI, recibe al menos dos archivos. Todos los archivos tienen el mismo prefijo de nombre de archivo, pero extensiones diferentes. Por ejemplo, las extensiones de los dos archivos que siempre recibirá son .shp y .shx.

Para recibir los archivos correspondientes a un conjunto de representaciones de formas, asigne el nombre que los archivos tienen en común al parámetro fileName. No especifique una extensión. De esta manera, asegura que se importen todos los archivos que necesita: los archivos .shp y .shx, y cualquier otro que se pueda incluir en la importación.

Por ejemplo, suponga que los archivos llamados Lakes.shp y Lakes.shx contienen un conjunto de representaciones de forma ESRI. Cuando importe estas representaciones, asignará únicamente el nombre ″Lakes″ al parámetro ″fileName″.

Los nombres de los archivos de transferencia SDE carecen de extensión. Por tanto, cuando importe un archivo de transferencia SDE, asignará su nombre, sin extensión, al parámetro ″fileName″.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización SYSADM o DBADM sobre la base de datos que contiene la tabla a la que se deben importar los datos de forma.
- Privilegio CONTROL sobre esta tabla.

Parámetros:

Tabla 81. Parámetros de entrada del 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 puede tener un valor nulo. Si no proporciona un valor para el parámetro layerSchema, se establecerá por omisión en el ID de usuario utilizado para invocar al procedimiento almacenado db2gse.gse_import_shape.
layerTable	VARCHAR(128)	Nombre de la tabla en la que se cargará el archivo de formas importado. Este parámetro no puede tener un valor nulo.
layerColumn	VARCHAR(30)	Nombre de la columna que se ha registrado como la capa en la que se deben cargar los datos de forma. Este parámetro no puede tener un valor nulo.
fileName	VARCHAR(128)	Nombre del archivo de formas que se debe importar. Este parámetro no puede tener un valor nulo.
exceptionFile	VARCHAR(128)	Nombre y vía de acceso del archivo donde se guardan las formas que no se pudieron importar. Este archivo se crea cuando se ejecuta el procedimiento almacenado db2gse.gse_import_shape. Asigne un nombre de archivo, sin extensión, al parámetro exceptionFile. Este parámetro no puede tener un valor nulo.
srId	INTEGER	Identificador de sistemas de referencia espacial que se debe utilizar para la capa en la que se deben cargar los datos de forma. Este parámetro puede tener un valor nulo. Si no se especifica este identificador, la transformación interna se establecerá en la máxima resolución posible para el archivo de formas.
commitScope	INTEGER	Número de registros por cada punto de control. Este parámetro puede tener un valor nulo.

Resultados:

Tabla 82. Parámetros de salida del procedimiento almacenado db2gse.gse_import_shape.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor de DB2 Spatial Extender.

db2gse.gse_register_gc

Utilice este procedimiento almacenado para registrar un geocodificador distinto del definido por omisión. Para determinar si un geocodificador ya está registrado, consulte la vista de catálogo DB2GSE.SPATIAL_GEOCODER.

Autorización:

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos donde reside el geocodificador que registra este procedimiento almacenado.

Parámetros:

Tabla 83. Parámetros de entrada del procedimiento almacenado db2gse.gse_register_gc.

Nombre	Tipo de datos	Descripción
gcId	INTEGER	Identificador numérico del geocodificador que se está registrando. Este identificador debe ser exclusivo dentro de la base de datos. Este parámetro no puede tener un valor nulo.
gcName	VARCHAR(64)	Descripción abreviada del geocodificador que se está registrando. Esta descripción debe estar formada por una cadena de caracteres que es exclusiva dentro de la base de datos. Este parámetro no puede tener un valor nulo.
vendorName	VARCHAR(64)	Nombre del proveedor que proporcionó el geocodificador que se está registrando. Este parámetro no puede tener un valor nulo.
primaryUDF	VARCHAR(256)	Nombre calificado al completo del geocodificador que se está registrando. Este parámetro no puede tener un valor nulo.

Tabla 83. Parámetros de entrada del procedimiento almacenado db2gse.gse_register_gc. (continuación)

Nombre	Tipo de datos	Descripción
precisionLevel	INTEGER	Grado de coincidencia que debe existir entre los datos de origen y los correspondientes datos de referencia para que el geocodificador procese los datos de origen satisfactoriamente. El nivel de precisión puede tener un valor comprendido entre el 1 y el 100 por ciento. Este parámetro no puede tener un valor nulo.
vendorSpecific	VARCHAR(256)	Información técnica proporcionada por el proveedor; por ejemplo, el nombre y vía de acceso de un archivo que el proveedor utiliza para definir parámetros. Este parámetro puede tener un valor nulo.
geoArea	VARCHAR(256)	Zona geográfica que se debe geocodificar. Este parámetro puede tener un valor nulo.
description	VARCHAR(256)	Comentarios proporcionados por el proveedor. Este parámetro puede tener un valor nulo.

Resultados:

Tabla 84. Parámetros de salida del procedimiento almacenado db2gse.gse_register_gc.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor de DB2 Spatial Extender.

db2gse.gse_register_layer

Utilice este procedimiento almacenado para registrar una columna espacial como capa. Cuando se procesa este procedimiento almacenado, la información sobre la capa que se está registrando se añade a la vista de catálogo DB2GSE.ST_GEOMETRY_COLUMNS.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Para una capa de tabla:
  - Autorización SYSADM o DBADM sobre la base de datos donde reside la tabla a la que pertenece la capa.
  - Privilegio CONTROL o ALTER sobre esta tabla.
- Para una capa de vista:

– Privilegio SELECT sobre la tabla o tablas base que contienen (1) los datos de dirección que se deben geocodificar para la capa y (2) los datos espaciales que resultan de la geocodificación.

Parámetros:

Tabla 85. Parámetros de entrada del 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 puede tener un valor nulo. Si no proporciona un valor para el parámetro layerSchema, se establecerá por omisión en el ID de usuario utilizado para invocar al procedimiento almacenado db2gse.gse_register_layer.
layerTable	VARCHAR(128)	Nombre de la tabla o vista donde reside la columna que se está registrando como capa. Este parámetro no puede tener un valor nulo.
layerColumn	VARCHAR(128)	Nombre de la columna que se está registrando como capa. Para una tabla, si la columna no existe, DB2 Spatial Extender la añadirá utilizando la sentencia ALTER. Para una vista, la columna debe existir previamente. Se puede especificar una sola columna para el parámetro layerColumn. Por tanto, cuando registra como capa varias columnas de una tabla o vista, debe ejecutar este procedimiento almacenado por separado para cada columna. Este parámetro no puede tener un valor nulo.
layerTypeName	VARCHAR(64)	Tipo de datos de la columna que se está registrando como capa. Sólo se aceptan tipos de datos proporcionados por DB2 Spatial Extender. Debe especificar el tipo de datos utilizando letras mayúsculas; por ejemplo: `ST_POINT` No es necesario especificar un nombre de esquema, pues se añade automáticamente. Este parámetro no puede ser nulo si la columna es una columna de tabla que se debe crear cuando se procese este procedimiento almacenado. En otro caso, si la columna ya existe dentro de una tabla o vista, este parámetro puede tener un valor nulo.
srId	INTEGER	Identificador de sistemas de referencia espacial utilizado para la capa. Este parámetro no puede tener un valor nulo para una capa de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra un capa de vista.

Tabla 85. Parámetros de entrada del procedimiento almacenado db2gse.gse_register_layer. (continuación)

Nombre	Tipo de datos	Descripción
geoSchema	VARCHAR(30)	Esquema de la tabla en la que está basada la vista a la que pertenece la columna. Este parámetro es aplicable cuando se registra como capa una columna de vista. Este parámetro puede ser nulo cuando se registra como capa una columna de vista. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de tabla. Las vistas basadas en más de una tabla base o en otras vistas no se pueden utilizar con este parámetro. Si no proporciona un valor para el parámetro geoSchema, toma por omisión el valor del parámetro layerSchema.
geoTable	VARCHAR(128)	Nombre de la tabla en la que está basada la vista a la que pertenece la columna. Este parámetro es aplicable cuando se registra como capa una columna de vista. Las vistas basadas en más de una tabla base o en otras vistas no se pueden utilizar con este parámetro. Este parámetro no puede ser nulo cuando se registra como capa una columna de vista. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de tabla.
geoColumn	VARCHAR(128)	Nombre de la columna de tabla en la que está basada la columna de vista. Este parámetro es aplicable cuando se registra como capa una columna de vista. Las vistas basadas en más de una tabla base o en otras vistas no se pueden utilizar con este parámetro. Este parámetro no puede ser nulo cuando se registra como capa una columna de vista. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de tabla.
nAttributes	SMALLINT	Número de columnas que contienen los datos fuente que se deben geocodificar para la capa. Este parámetro puede ser nulo cuando se registra como capa una columna de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de vista.

Tabla 85. Parámetros de entrada del procedimiento almacenado db2gse.gse_register_layer. (continuación)

Nombre	Tipo de datos	Descripción
attr1Name	VARCHAR(128)	Nombre de la primera columna que contiene los datos fuente que se deben geocodificar para la capa. Este parámetro puede ser nulo cuando se registra como capa una columna de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de vista. Si desea utilizar el geocodificador por omisión, deberá guardar las direcciones postales en la columna attr1Name.
attr2Name	VARCHAR(128)	Nombre de la segunda columna que contiene los datos fuente que se deben geocodificar para la capa. Este parámetro puede ser nulo cuando se registra como capa una columna de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de vista. Si desea utilizar el geocodificador por omisión, deberá 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 la capa. Este parámetro puede ser nulo cuando se registra como capa una columna de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de vista. Si desea utilizar el geocodificador por omisión, deberá guardar los nombres o abreviaturas de estados o provincias en la columna attr3Name.
attr4Name	VARCHAR(128)	Nombre de la cuarta columna que contiene los datos fuente que se deben geocodificar para la capa. Este parámetro puede ser nulo cuando se registra como capa una columna de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de vista. Si desea utilizar el geocodificador por omisión, deberá guardar los códigos postales en la columna attr4Name.
attr5Name	VARCHAR(128)	Nombre de la quinta columna que contiene los datos fuente que se deben geocodificar para la capa. Este parámetro puede ser nulo cuando se registra como capa una columna de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de vista. El geocodificador por omisión no tiene en cuenta la columna Attr5Name.

Tabla 85. Parámetros de entrada del procedimiento almacenado db2gse.gse_register_layer. (continuación)

Nombre	Tipo de datos	Descripción
attr6Name	VARCHAR(128)	Nombre de la sexta columna que contiene los datos fuente que se deben geocodificar para la capa. Este parámetro puede ser nulo cuando se registra como capa una columna de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de vista. El geocodificador por omisión no tiene en cuenta la columna Attr6Name.
attr7Name	VARCHAR(128)	Nombre de la séptima columna que contiene los datos fuente que se deben geocodificar para la capa. Este parámetro puede ser nulo cuando se registra como capa una columna de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de vista. El geocodificador por omisión no tiene en cuenta la columna Attr7Name.
attr8Name	VARCHAR(128)	Nombre de la octava columna que contiene los datos fuente que se deben geocodificar para la capa. Este parámetro puede ser nulo cuando se registra como capa una columna de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de vista. El geocodificador por omisión no tiene en cuenta la columna Attr8Name.
attr9Name	VARCHAR(128)	Nombre de la novena columna que contiene los datos fuente que se deben geocodificar para la capa. Este parámetro puede ser nulo cuando se registra como capa una columna de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de vista. El geocodificador por omisión no tiene en cuenta la columna Attr9Name.
attr10Name	VARCHAR(128)	Nombre de la décima columna que contiene los datos fuente que se deben geocodificar para la capa. Este parámetro puede ser nulo cuando se registra como capa una columna de tabla. DB2 Spatial Extender no tiene en cuenta este parámetro cuando se registra como capa una columna de vista. El geocodificador por omisión no tiene en cuenta la columna Attr10Name.

Resultados:

Tabla 86. Parámetros de salida del procedimiento almacenado db2gse.gse_register_layer.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor de DB2 Spatial Extender.

Restricciones:

Este procedimiento almacenado no es efectivo para los tipos de tabla siguientes:

- A = Alias
- H = Tabla de jerarquía
- N = Apodo
- S = Tabla de resumen
- U = Tabla tipificada
- W = Vista tipificada

También rigen las restricciones siguientes:

- Si está registrando como capa una columna de vista, debe estar basada en una columna de tabla que ya se ha registrado como capa.
- Los datos que se deben geocodificar para la capa que está registrando pueden estar contenidos en un máximo de diez columnas de atributos.

db2gse.gse_run_gc

Utilice este procedimiento almacenado para ejecutar un geocodificador en la modalidad de proceso por lotes.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Autorización SYSADM o DBADM sobre la base de datos donde reside la tabla sobre la que actuará el geocodificador especificado.
- Privilegio CONTROL o UPDATE sobre esta tabla.

Parámetros:

Tabla 87. Parámetros de entrada del 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 puede tener un valor nulo. Si no proporciona un valor para el parámetro layerSchema, se establecerá por omisión en el ID de usuario utilizado para invocar al procedimiento almacenado db2gse.gse_run_gc.

Tabla 87. Parámetros de entrada del procedimiento almacenado db2gse.gse_run_gc. (continuación)

Nombre	Tipo de datos	Descripción
layerTable	VARCHAR(128)	Nombre de la tabla donde reside la columna a la que se añadirán los datos geocodificados. Este parámetro no puede tener un valor nulo.
layerColumn	VARCHAR(128)	Nombre de la columna a la que se añadirán los datos geocodificados. Este parámetro no puede tener un valor nulo.
gcId	INTEGER	Identificador del geocodificador que se desea utilizar. Este parámetro puede tener un valor nulo. Para conocer los identificadores de los geocodificadores registrados, consulte la vista de catálogo DB2GSE.SPATIAL_GEOCODER.
precisionLevel	INTEGER	Grado de coincidencia que debe existir entre los datos de origen y los correspondientes datos de referencia para que el geocodificador procese los datos de origen satisfactoriamente. Este parámetro puede tener un valor nulo. El nivel de precisión puede tener un valor comprendido entre el 1 y el 100 por ciento.
vendorSpecific	VARCHAR(256)	Información técnica proporcionada por el proveedor; por ejemplo, el nombre y vía de acceso de un archivo que el proveedor utiliza para definir parámetros. Este parámetro puede tener un valor nulo.
whereClause	VARCHAR(256)	Es el cuerpo de la cláusula WHERE. Define una restricción sobre el conjunto de registros que se deben geocodificar. La cláusula puede especificar cualquier columna de atributos de la tabla sobre la que debe actuar el geocodificador. Este parámetro puede tener un valor nulo.
commitScope	INTEGER	Número de registros por cada punto de control. Este parámetro puede tener un valor nulo.

Resultados:

Tabla 88. Parámetros de salida del procedimiento almacenado db2gse.gse_run_gc.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor de DB2 Spatial Extender.

db2gse.gse_unregist_gc

Utilice este procedimiento almacenado para desregistrar un geocodificador distinto del definido por omisión. Para buscar información sobre el geocodificador que desea desregistrar, consulte la vista de catálogo DB2GSE.SPATIAL_GEOCODER.

Autorización:

El ID de usuario utilizado para invocar a este procedimiento almacenado debe tener autorización SYSADM o DBADM sobre la base de datos donde reside el geocodificador que se debe desregistrar.

Parámetros::

Tabla 89. Parámetro de entrada del procedimiento almacenado db2gse.gse_unregist_gc.

Nombre	Tipo de datos	Descripción
gcId	INTEGER	Identificador del geocodificador que se debe desregistrar. Este parámetro no puede tener un valor nulo.

Resultados:

Tabla 90. Parámetros de salida del procedimiento almacenado db2gse.gse_unregist_gc.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor de DB2 Spatial Extender.

db2gse.gse_unregist_layer

Utilice este procedimiento almacenado para desregistrar una capa. El procedimiento almacenado desregistra la capa haciendo lo siguiente:

- Elimina la definición de la capa en las tabla de catálogo de DB2 Spatial Extender.
- Suprime la restricción de comprobación que DB2 Spatial Extender definió sobre la tabla base de la capa para asegurar que los datos espaciales de la capa cumplan los requisitos del sistema de referencia espacial de la capa.
- Descarta los desencadenantes que se utilizan para actualizar la columna espacial cada vez que se añaden, modifican o eliminan datos sobre direcciones.

Cuando se geocodifican los datos sobre direcciones contenidos en una fila de tabla, los datos espaciales resultantes se colocan en la misma fila. Por tanto, si se suprime la fila, se suprimen al mismo tiempo los datos sobre direcciones y los datos espaciales. Los desencadenantes no suprimen los datos espaciales. Cuando se procesa este procedimiento almacenado, la información sobre la capa se elimina de la vista de catálogo DB2GSE.GEOMETRY_COLUMNS.

Autorización:

El ID de usuario utilizado para invocar al procedimiento almacenado debe tener las autorizaciones o privilegios siguientes:

- Para una capa de tabla:
  - Autorización SYSADM o DBADM sobre la base de datos donde reside la tabla base de la capa.
  - Privilegio CONTROL o ALTER sobre esta tabla.
- Para una capa de vista:
  - Privilegio SELECT sobre la tabla o tablas base que contienen (1) los datos de dirección que se deben geocodificar para la capa y (2) los datos espaciales que resultan de la geocodificación.

Parámetros:

Tabla 91. Parámetros de entrada del 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 puede tener un valor nulo. Si no proporciona un valor para el parámetro layerSchema, se establecerá por omisión en el ID de usuario utilizado para invocar al procedimiento almacenado db2gse.gse_unregister_layer. Debe utilizar letras mayúsculas para especificar cualquier nombre de esquema, tabla, vista, columna o de capa que asigne a un parámetro.
layerTable	VARCHAR(128)	Nombre de la tabla donde reside la columna especificada en el parámetro layerColumn. Este parámetro no puede tener un valor nulo.
layerColumn	VARCHAR(128)	Nombre de la columna espacial definida como capa que desea desregistrar. Este parámetro no puede tener un valor nulo. Se puede especificar una sola capa para el parámetro layerColumn. Por tanto, cuando desregistra varias capas de una tabla o vista, debe ejecutar este procedimiento almacenado por separado para cada capa.

Resultados:

Tabla 92. Parámetros de salida del procedimiento almacenado db2gse.gse_unregist_layer.

Nombre	Tipo de datos	Descripción
msgCode	INTEGER	Código asociado a los mensajes que pueden ser emitidos por quien llama a este procedimiento almacenado.
msgText	VARCHAR(1024)	Mensaje de error completo, tal como se crea en el servidor de DB2 Spatial Extender.

Restricción::

Si una columna de vista que se ha definido como capa de vista está basada en una columna de tabla que se ha definido como capa de tabla, no puede desregistrar esta capa de tabla hasta que desregistre la capa de vista.

Índice

Apéndice B. Vistas de catálogo obsoletas

Este tema describe las vistas de catálogo obsoletas.

Nota: Recomendación: cree todas la nuevas aplicaciones con las vistas definidas en DB2 Spatial Extender Versión 8. Actualice las aplicaciones existentes para que utilicen las vistas definidas en la Versión 8. Las aplicaciones que hagan referencia a las tablas de catálogo no documentadas definidas en la Versión 7 ya no serán efectivas después de migrar a la Versión 8 y se deben modificar para utilizar las vistas de catálogo documentadas de la Versión 8.

DB2GSE.COORD_REF_SYS

Cuando se habilita una base de datos para operaciones espaciales, DB2 Spatial Extender registra en una tabla de catálogo los sistemas de coordenadas que el usuario puede utilizar. Determinadas columnas de esa tabla de catálogo forman la vista de catálogo DB2GSE.COORD_REF_SYS, que se describe en la tabla siguiente.

Tabla 93. Columnas de la vista de catálogo DB2GSE.COORD_REF_SYS

Nombre	Tipo de datos	Posibilidad de contener nulos	Contenido
CSID	INTEGER	Sí	Identificador numérico exclusivo del sistema de coordenadas. Si el sistema de coordenadas se creó utilizando la interfaz de administración de la Versión 8, no se registrará ningún CSID y en su lugar se utilizará un valor nulo.
CS_NAME	VARCHAR(64)	No	Nombre de este sistema de coordenadas.
AUTH_NAME	VARCHAR(256)	Sí	Nombre de la organización que creó el sistema de coordenadas; por ejemplo, European Petroleum Survey Group (EPSG).
AUTH_SRID	INTEGER	Sí	Identificador numérico que la organización especificada en la columna AUTH_NAME asigna al sistema de coordenadas.
DESC	VARCHAR(256)	Sí	Descripción del sistema de coordenadas.
SRTEXT	VARCHAR(2048)	No	Comentarios sobre el sistema de coordenadas.

DB2GSE.GEOMETRY_COLUMNS

Cuando el usuario crea una capa, DB2 Spatial Extender registra el identificador de la capa e información sobre ella en una tabla de catálogo. Determinadas columnas de esa tabla de catálogo forman la vista de catálogo DB2GSE.GEOMETRY_COLUMNS, que se describe en la tabla siguiente.

Vista de catálogo obsoletas

Tabla 94. Columnas de la vista de catálogo DB2GSE.GEOMETRY_COLUMNS

Nombre	Tipo de datos	Posibilidad de contener nulos	Contenido
LAYER_CATALOG	VARCHAR(30)	Sí	NULL. El concepto LAYER_CATALOG no existe en DB2 Spatial Extender.
LAYER_SCHEMA	VARCHAR(30)	No	Esquema de la tabla o vista donde reside la columna que se registró como capa.
LAYER_TABLE	VARCHAR(128)	No	Nombre de la tabla o vista donde reside la columna que se registró como capa.
LAYER_COLUMN	VARCHAR(128)	No	Nombre de la columna que se registró como capa.
GEOMETRY_TYPE	INTEGER	Sí	Tipo de datos de la columna que se registró como capa. Si la columna tiene un subtipo definido por el usuario perteneciente a cualquiera de los tipos de geometría definidos por Spatial Extender, este valor será nulo.
SRID	INTEGER	No	Identificador de sistemas de referencia espacial utilizado para los valores de la columna que se registró como capa.
STORAGE_TYPE	INTEGER	Sí	NULL.

DB2GSE.SPATIAL_GEOCODER

Los geocodificadores disponibles están registrados en una tabla de catálogo. Determinadas columnas de esa tabla de catálogo forman la vista de catálogo DB2GSE.SPATIAL_GEOCODER, que se describe en la tabla siguiente.

Tabla 95. Columnas de la vista de catálogo DB2GSE.SPATIAL_GEOCODER

Nombre	Tipo de datos	Posibilidad de contener nulos	Contenido
GCID	INTEGER	No	Identificador numérico del geocodificador.
GC_NAME	VARCHAR(64)	No	Nombre identificador del geocodificador.
VENDOR_NAME	VARCHAR(128)	No	Nombre del proveedor que proporcionó el geocodificador.
PRIMARY_UDF	VARCHAR(256)	No	Nombre calificado al completo del geocodificador.
PRECISION_LEVEL	INTEGER	No	Grado de coincidencia que debe existir entre los datos fuente y los correspondientes datos de referencia para poder ser procesados satisfactoriamente por el geocodificador.
VENDOR_SPECIFIC	VARCHAR(256)	Sí	Nombre y vía de acceso de un archivo que puede ser utilizado por un proveedor para definir parámetros especiales que puedan ser utilizados por el geocodificador.
GEO_AREA	VARCHAR(256)	Sí	Zona geográfica que contiene las ubicaciones que se deben geocodificar.
DESCRIPTION	VARCHAR(256)	Sí	Descripción del geocodificador.

DB2GSE.SPATIAL_REF_SYS

Cuando el usuario crea un sistema de referencia espacial, DB2 Spatial Extender registra el identificador de sistemas de referencia espacial e información sobre el mismo en una tabla de catálogo. Determinadas columnas de esa tabla de catálogo forman la vista de catálogo DB2GSE.SPATIAL_REF_SYS, que se describe en la tabla siguiente.

Tabla 96. Columnas de la vista de catálogo DB2GSE.SPATIAL_REF_SYS

Nombre	Tipo de datos	Posibilidad de contener nulos	Contenido
SRID	INTEGER	No	Identificador, definido por el usuario, correspondiente al sistema de referencia espacial.
SR_NAME	VARCHAR(64)	No	Nombre del sistema de referencia espacial.
CSID	INTEGER	No	Identificador numérico del sistema de coordenadas en el que se basa el sistema de referencia espacial.
CS_NAME	VARCHAR(64)	No	Nombre del sistema de coordenadas en el que se basa el sistema de referencia espacial.
AUTH_NAME	VARCHAR(256)	Sí	Nombre de la organización que define las normas para el sistema de referencia espacial.
AUTH_SRID	INTEGER	Sí	Identificador que la organización especificada en la columna AUTH_NAME asigna al sistema de referencia espacial.
SRTEXT	VARCHAR(2048)	No	Comentarios sobre el sistema de referencia espacial.
FALSEX	FLOAT	No	Número que se resta de un valor de coordenada X negativo y da como resultado un número positivo o el valor 0.
FALSEY	FLOAT	No	Número que se resta de un valor de coordenada Y negativo y da como resultado un número positivo o el valor 0.
XYUNITS	FLOAT	No	Número que se multiplica por un valor decimal de coordenada X o Y y da como resultado un valor entero que se puede guardar como dato de 32 bits.
FALSEZ	FLOAT	No	Número que se resta de un valor de coordenada Z negativo y da como resultado un número positivo o el valor 0.
ZUNITS	FLOAT	No	Número que se multiplica por un valor decimal de coordenada Z y da como resultado un valor entero que se puede guardar como dato de 32 bits.
FALSEM	FLOAT	No	Número que se resta de un valor de medida negativo y da como resultado un número positivo o el valor 0.
MUNITS	FLOAT	No	Número que se multiplica por un valor decimal de medida y da como resultado un valor entero que se puede guardar como dato de 32 bits.

Vista de catálogo obsoletas

Apéndice C. Funciones espaciales obsoletas

Este tema describe las funciones que han dejado de utilizarse. La tabla siguiente lista todas las funciones espaciales obsoletas y las nuevas funciones que ahora se utiliza para la Versión 8.

Tabla 97. Funciones obsoletas y sus nuevos equivalentes.

Función obsoleta	Nueva función
AsShape	ST_AsShape
GeometryFromShape	ST_Geometry
Is3D	ST_Is3D
IsMeasured	ST_IsMeasured
LineFromShape	ST_Linestring
LocateAlong	ST_FindMeasure
LocateBetween	ST_MeasureBetween
M	ST_M
MLine FromShape	ST_MultiLineString
MPointFromShape	ST_MultiPoint
MPolyFromShape	ST_MultiPolygon
PointFromShape	ST_Point
PolyFromShape	ST_Polygon
ShapeToSQL	ST_Geometry
ST_GeomFromText	ST_Geometry
ST_GeomFromWKB	ST_Geometry
ST_LineFromText	ST_LineString
ST_LineFromWKB	ST_LineString
ST_MLineFromText	ST_MultiLineString
ST_MLineFromWKB	ST_MultiLineString
ST_MPointFromText	ST_MultiPoint
ST_MPointFromWKB	ST_MultiPoint
ST_MPolyFromText	ST_MultiPolygon
ST_MPolyFromWKB	ST_MultiPolygon
ST_OrderingEquals	
ST_Point(Double, Double, db2gse.coordref)	ST_Point(Double, Double, Integer)
ST_PointFromText	ST_Point
ST_PolyFromText	ST_Polygon
ST_PolyFromWKB	ST_Polygon
ST_Transform(Double, Double, db2gse.coordref)	ST_Transform(ST_Geometry, Integer)
ST_SymmetricDiff	ST_SymDifference
Z	ST_Z

AsShape

Finalidad:

AsShape toma un objeto de geometría como entrada y devuelve un valor BLOB.

Formato:

db2gse.AsShape(g db2gse.ST_Geometry)

Resultados:

BLOB(1m)

GeometryFromShape

Finalidad:

GeometryFromShape toma como entrada una forma y un identificador de sistemas de referencia espacial y devuelve un objeto de geometría.

Formato:

db2gse.GeometryFromShape(ShapeGeometry Blob(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_Geometry

Is3d

Finalidad:

Is3d toma un objeto de geometría como entrada y devuelve un 1 si el objeto tiene coordenadas 3D; en otro caso, devuelve un 0.

Formato:

db2gse.Is3d(g db2gse.ST_Geometry)

Resultados:

Integer

IsMeasured

Finalidad:

IsMeasured toma un objeto de geometría como entrada y devuelve un 1 si el objeto tiene medidas; en otro caso, devuelve un 0.

Formato:

db2gse.IsMeasured(g db2gse.ST_Geometry)

Resultados:

Integer

LineFromShape

Finalidad:

LineFromShape toma como entrada una forma de tipo punto y un identificador de sistemas de referencia espacial y devuelve una cadena lineal.

Formato:

db2gse.Line FromShape(ShapeLineString Blob(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_LineString

LocateAlong

Finalidad:

LocateAlong toma como entrada un objeto de geometría y una medida y devuelve, en forma de multipunto, el conjunto de puntos encontrados en la posición indicada por la medida.

Si LocateAlong recibe como entrada un multipunto y una medida, y el multipunto no incluye a esta medida, entonces LocateAlong devuelve un punto vacío (POINT EMPTY).

Formato:

db2gse.LocateAlong(g db2gse.ST_Geometry, measure Double)

Resultados:

db2gse.ST_Geometry

LocateBetween

Finalidad:

LocateBetween toma como entrada un objeto de geometría y dos ubicaciones indicadas por una medida y devuelve una geometría que representa el conjunto de trayectos discontinuos entre las dos ubicaciones.

Formato:

db2gse.LocateBetween(g db2gse.ST_Geometry, measure Double, measure Double)

Resultados:

db2gse.ST_Geometry

M

Finalidad:

M toma un punto como entrada y devuelve su medida.

Formato:

db2gse.M(p db2gse.ST_Point)

Resultados:

Double

MLine FromShape

Finalidad:

MLine FromShape toma como entrada una forma de tipo multilínea y un identificador de sistemas de referencia espacial y devuelve una multilínea.

Formato:

db2gse.MLineFromShape(ShapeMultiLineString Blob(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_MultiLineString

MPointFromShape

Finalidad:

MPointFromShape toma como entrada una forma de tipo multipunto y un identificador de sistemas de referencia espacial y devuelve un multipunto.

Formato:

db2gse.MPointFromShape(ShapeMultiPoint BLOB(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_MultiPoint

MPolyFromShape

Finalidad:

MPolyFromShape toma como entrada una forma de tipo multipolígono y un identificador de sistemas de referencia espacial y devuelve un multipolígono.

Formato:

db2gse.MPolyFromShape(ShapeMultiPolygon Blob(1m), SRID db2gse.coordref)

Resultados:

db2gse.ST_MultiPolygon

PointFromShape

Finalidad:

PointFromShape toma como entrada una forma de tipo punto y un identificador de sistemas de referencia espacial y devuelve un punto.

Formato:

db2gse.PointFromShape(db2gse.ShapePoint blob(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_Point

PolyFromShape

Finalidad:

PolyFromShape toma como entrada una forma de tipo polígono y un identificador de sistemas de referencia espacial y devuelve un polígono.

Formato:

db2gse.PolyFromShape (ShapePolygon Blob(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_Polygon

ShapeToSQL

Finalidad:

ShapeToSQL crea un valor db2gse.ST_Geometry a partir de su representación de forma. Se utiliza automáticamente un valor de SRID igual a 0.

Formato:

db2gse.ShapeToSQL(ShapeGeometry blob(1M))

Resultados:

db2gse.ST_Geometry

ST_GeomFromText

Finalidad:

ST_GeomFromText toma como entrada una representación de texto convencional y un identificador de sistemas de referencia espacial y devuelve un objeto de geometría.

Formato:

db2gse.ST_GeomFromText(geometryTaggedText Varchar(4000), SRID db2gse.coordref)

Resultados:

db2gse.ST_Geometry

ST_GeomFromWKB

Finalidad:

ST_GeomFromWKB toma como entrada una representación binaria convencional y un identificador de sistemas de referencia espacial y devuelve un objeto de geometría.

Formato:

db2gse.ST_GeomFromWKB(WKBGeometry Blob(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_Geometry

ST_LineFromText

Finalidad:

ST_LineFromText toma como entrada una representación de texto convencional de tipo cadena lineal y un identificador de sistemas de referencia espacial y devuelve una cadena lineal.

Formato:

db2gse.ST_LineFromText(lineStringTaggedText Varchar(4000), SRID db2gse.coordref)

Resultados:

db2gse.ST_LineString

ST_LineFromWKB

Finalidad:

ST_LineFromWKB toma como entrada una representación binaria convencional de tipo cadena lineal y un identificador de sistemas de referencia espacial y devuelve una cadena lineal.

Formato:

db2gse.ST_LineFromWKB(WKBLineString Blob(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_LineString

ST_MLineFromText

Finalidad:

ST_MLineFromText toma como entrada una representación de texto convencional de tipo multilínea y un identificador de sistemas de referencia espacial y devuelve una multilínea.

Formato:

db2gse.ST_MLineFromText(multiLineStringTaggedText String, SRID db2gse.coordref)

Resultados:

db2gse.ST_MultiLineString

ST_MLineFromWKB

Finalidad:

ST_MLineFromWKB toma como entrada una representación binaria convencional de tipo multilínea y un identificador de sistemas de referencia espacial y devuelve una multilínea.

Formato:

db2gse.ST_MLineFromWKB(WKBMultiLineString Blob(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_MultiLineString

ST_MPointFromText

Finalidad:

ST_MPointFromText toma como entrada una representación de texto convencional de tipo multipunto y un identificador de sistemas de referencia espacial y devuelve un multipunto.

Formato:

db2gse.ST_MPointFromText(multiPointTaggedText Varchar(4000), SRID db2gse.coordref)

Resultados:

db2gse.ST_MultiPoint

ST_MPointFromWKB

Finalidad:

ST_MPointFromWKB toma como entrada una representación binaria convencional de tipo multipunto y un identificador de sistemas de referencia espacial y devuelve un multipunto.

Formato:

db2gse.ST_MPointFromWKB(WKBMultiPoint Blob(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_MultiPoint

ST_MPolyFromText

Finalidad:

ST_MPolyFromText toma como entrada una representación de texto convencional de tipo multipolígono y un identificador de sistemas de referencia espacial y devuelve un multipolígono.

Esta función no puede tomar como entrada un multipolígono que contenga varios polígonos con las mismas coordenadas.

Formato:

db2gse.ST_MPolyFromText(multiPolygonTaggedText Varchar(4000), SRID db2gse.coordref)

Resultados:

db2gse.ST_MultiPolygon

ST_MPolyFromWKB

Finalidad:

ST_MPolyFromWKB toma como entrada una representación binaria convencional de tipo multipolígono y un identificador de sistemas de referencia espacial y devuelve un multipolígono.

Formato:

db2gse.ST_MPolyFromWKB(WKBMultiPolygon Blob(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_MultiPolygon

ST_OrderingEquals

Finalidad:

ST_OrderingEquals compara dos geometrías y devuelve un 1 (TRUE) si las geometrías son iguales y las coordenadas están en el mismo orden; en otro caso, devuelve un 0 (FALSE).

Formato:

db2gse.ST_OrderingEquals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Resultados:

Integer

ST_Point

Finalidad:

ST_Point devuelve un ST_Point, a partir de unas coordenadas X,Y y el sistema de referencia espacial.

Formato:

db2gse.ST_Point(X Double, Y Double, SRID db2gse.coordref)

Resultados:

db2gse.ST_Point

ST_PointFromText

Finalidad:

ST_PointFromText toma como entrada una representación de texto convencional de tipo punto y un identificador de sistemas de referencia espacial y devuelve un punto.

Formato:

db2gse.ST_PointFromText(pointTaggedText Varchar(4000), SRID db2gse.coordref)

Resultados:

db2gse.ST_Point

ST_PolyFromText

Finalidad:

ST_PolyFromText toma como entrada una representación de texto convencional de tipo polígono y un identificador de sistemas de referencia espacial y devuelve un polígono.

Formato:

db2gse.ST_PolyFromText(polygonTaggedText Varchar(4000), SRID db2gse.coordref)

Resultados:

db2gse.ST_Polygon

ST_PolyFromWKB

Finalidad:

ST_PolyFromWKB toma como entrada una representación binaria convencional de tipo polígono y un identificador de sistemas de referencia espacial y devuelve un polígono.

Formato:

db2gse.ST_PolyFromWKB(WKBPolygon Blob(1M), SRID db2gse.coordref)

Resultados:

db2gse.ST_Polygon

ST_Transform

Finalidad:

ST_Transform asocia una geometría a un sistema de referencia espacial distinto de aquél al cual está asociado actualmente la geometría.

Formato:

db2gse.ST_Transform(g db2gse.ST_Geometry, SRID db2gse.coordref)

Resultados:

db2gse.ST_Geometry

ST_SymmetricDiff

Finalidad:

ST_SymmetricDiff toma como entrada dos objetos de geometría y devuelve un objeto de geometría que es la diferencia simétrica de los objetos originales.

La función ST_SymmetricDiff devuelve la diferencia simétrica (el equivalente espacial de la operación lógica XOR) de dos geometrías que forman intersección y tienen la misma dimensión. Si estas dos geometrías son iguales, ST_SymmetricDiff devuelve una geometría vacía. Si las geometrías no son iguales, entonces una parte de una o ambas geometrías quedará fuera del área de intersección. ST_SymmetricDiff devuelve en forma de colección la porción o porciones que no se solapan; por ejemplo, en forma de multipolígono.

Si ST_SymmetricDiff recibe como entrada geometrías de dimensiones diferentes, el resultado devuelto es un valor nulo.

Formato:

db2gse.ST_SymmetricDiff(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

Resultados:

db2gse.ST_Geometry

Z

Finalidad:

Z toma un punto como entrada y devuelve su coordenada Z.

Formato:

db2gse.Z(p db2gse.ST_Point)

Resultados:

Double

Información relacionada:

- “ST_AsShape” en la página 362
- “ST_MeasureBetween, ST_LocateBetween” en la página 445
- “ST_EnvIntersects” en la página 392
- “ST_FindMeasure o ST_LocateAlong” en la página 398
- “ST_Geometry” en la página 406
- “ST_Is3d” en la página 419
- “ST_LineString” en la página 432
- “ST_M” en la página 434
- “ST_MultiLineString” en la página 462
- “ST_MultiPoint” en la página 464
- “ST_MultiPolygon” en la página 465

Funciones espaciales obsoletas

Avisos

Es posible que IBM no comercialice en todos los países algunos productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en tramitación que afecten al tema tratado en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE.UU.

Para realizar consultas sobre licencias referentes a información de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país/región o escribir a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokio 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país/región en donde tales disposiciones sean incompatibles con la legislación local:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener inexactitudes técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán a las nuevas ediciones de la publicación. IBM puede efectuar, en cualquier momento y sin previo aviso, mejoras y cambios en los productos y programas descritos en esta publicación.

Las referencias hechas en esta publicación a sitios Web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen un aval de esos

sitios Web. La información contenida en esos sitios Web no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de dichos sitios Web.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información sobre él con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido éste) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADÁ

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos el pago de una tarifa.

El programa bajo licencia descrito en este documento y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Acuerdo de Cliente de IBM, el Acuerdo Internacional de Programas Bajo Licencia de IBM o cualquier acuerdo equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado en sistemas experimentales y no es seguro que estas mediciones sean las mismas en los sistemas disponibles comercialmente. Además, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios del presente manual deben verificar los datos aplicables para su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni ninguna otra afirmación referente a productos que no son de IBM. Las preguntas sobre las prestaciones de productos que no son de IBM deben dirigirse a los proveedores de esos productos.

Todas las declaraciones de intenciones de IBM están sujetas a cambio o cancelación sin previo aviso, y sólo representan objetivos.

Este manual puede contener ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente fortuita.

LICENCIA DE COPYRIGHT:

Este manual puede contener programas de aplicaciones de ejemplo escritos en lenguaje fuente, que muestran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo como desee, sin pago alguno a IBM, con la intención de desarrollar, utilizar, comercializar o distribuir programas de aplicaciones de acuerdo con la interfaz de programación de aplicaciones correspondiente a la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por lo tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas.

Cada copia o parte de estos programas de ejemplo o cualquier trabajo derivado debe incluir una nota de copyright como la siguiente:

© (*nombre de la empresa*) (*año*). Partes de este código proceden de programas de ejemplo de IBM Corp. © Copyright IBM Corp. *_entre el o los años_*. Reservados todos los derechos.

Marcas registradas

Los términos siguientes son marcas registradas de International Business Machines Corporation en los EE.UU. y/o en otros países y se han utilizado como mínimo en uno de los documentos de la biblioteca de documentación de DB2 UDB.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeriesLAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

Los términos siguientes son marcas registradas de otras empresas y se han utilizado como mínimo en uno de los documentos de la biblioteca de documentación de DB2 UDB:

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los EE.UU. y/o en otros países.

Intel y Pentium son marcas registradas de Intel Corporation en los EE.UU. y/o en otros países.

Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en los EE.UU. y/o en otros países.

UNIX es marca registrada de The Open Group en los EE.UU. y/o en otros países.

Otros nombres de empresas, productos o servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

E

F

H

I

L

M

N

P

R

S

T

U

V

W

Z

Cómo ponerse en contacto con IBM

En los EE.UU., puede ponerse en contacto con IBM llamando a uno de los siguientes números:

- 1-800-IBM-SERV (1-800-426-7378) para servicio al cliente
- 1-888-426-4343 para obtener información sobre las opciones de servicio técnico disponibles
- 1-800-IBM-4YOU (426-4968) para márketing y ventas de DB2

En Canadá, puede ponerse en contacto con IBM llamando a uno de los siguientes números:

- 1-800-IBM-SERV (1-800-426-7378) para servicio al cliente
- 1-800-465-9600 para obtener información sobre las opciones de servicio técnico disponibles
- 1-800-IBM-4YOU (1-800-426-4968) para márketing y ventas de DB2

Para localizar una oficina de IBM en su país o región, consulte IBM Directory of Worldwide Contacts en el sitio Web http://www.ibm.com/planetwide

Información sobre productos

La información relacionada con productos DB2 Universal Database se encuentra disponible por teléfono o a través de la World Wide Web en el sitio http://www.ibm.com/software/data/db2/udb

Este sitio contiene la información más reciente sobre la biblioteca técnica, pedidos de manuales, descargas de productos, grupos de noticias, FixPaks, novedades y enlaces con recursos de la Web.

Si vive en los EE.UU., puede llamar a uno de los números siguientes:

- 1-800-IBM-CALL (1-800-426-2255) para solicitar productos u obtener información general.
- 1-800-879-2755 para solicitar publicaciones.

Para obtener información sobre cómo ponerse en contacto con IBM desde fuera de los EE.UU., vaya a la página IBM Worldwide en el sitio www.ibm.com/planetwide

SC10-3755-01

Spine information:

IBM® DB2® Spatial Extender y Geodetic Extender

DB2 Spatial Extender y Geodetic Extender Guía del usuario y manual de consulta

Versión 8.2