IBM DB2 10.1
para Linux, UNIX e Windows

Referência e Guia do Usuário do Spatial Extender

S517-0064-00

IBM DB2 10.1
para Linux, UNIX e Windows

Referência e Guia do Usuário do Spatial Extender

S517-0064-00

Note
Before using this information and the product it supports, read the general information under Apêndice B, "Avisos", na página 451.

Edition Notice

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at http://www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at http://www.ibm.com/planetwide/

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Índice

Capítulo 18. Funções Espaciais . . . 227

Capítulo 1. A Finalidade do DB2 Spatial Extender

Utilize o DB2 Spatial Extender para gerar e analisar informações espaciais sobre recursos geográficos e para armazenar e gerenciar os dados nos quais estas informações são baseadas. Um recurso geográfico (às vezes chamado de recurso nesta discussão, para abreviar) é algo no mundo real que tem uma localização identificável, ou algo que pode ser imaginado como existente em uma localização identificável. Um recurso pode ser:

- Um objeto (ou seja, uma entidade concreta de qualquer tipo); por exemplo, um rio, uma floresta ou uma sequência de montanhas.
- Um espaço, uma zona de segurança em torno de um local perigoso, ou uma área de marketing atendida por um determinado ramo de negócios.
- Um evento que ocorre em uma localização que pode ser definida; por exemplo, um acidente automobilístico que ocorreu em um determinado cruzamento, ou uma transação de vendas em uma loja específica.

Existem recursos em vários ambientes. Por exemplo, os objetos mencionados na lista anterior — rio, floresta, sequência de montanhas — pertencem ao ambiente natural. Outros objetos, como cidades, edifícios e escritórios pertencem ao ambiente cultural. Ainda existem outros, como parques, zoológicos e zonas rurais que representam uma combinação dos ambientes natural e cultural.

Nesta discussão, o termo informações espaciais refere-se ao tipo de informação que o DB2 Spatial Extender disponibiliza para seus usuários, ou seja, fatos e figuras sobre as localizações de recursos geográficos. Exemplos de informações espaciais são:

- Localizações de recursos geográficos no mapa (por exemplo, valores longitudinais e latitudinais que definem onde as cidades estão situadas)
- A localização de recursos geográficos em relação um ao outro (por exemplo, pontos dentro de uma cidade onde hospitais e clínicas estão localizados, ou a proximidade das residências da cidade em relação a zonas de terremoto)
- Modos como os recursos geográficos estão relacionados entre si (por exemplo, informações de que um determinado sistema fluvial está contido dentro de um região específica, ou de que determinadas pontes naquela região atravessam os braços do sistema fluvial)
- Medidas que se aplicam a um ou mais recursos geográficos (por exemplo, a distância entre um prédio de escritórios e sua divisão de terreno ou o comprimento de um perímetro de preservação de um pássaro)

Informações espaciais, isoladas ou em conjunto com dados relacionais tradicionais, podem ajudá-lo nas atividades como definir as áreas nas quais você oferece serviços e determinar localizações de possíveis mercados. Suponha, por exemplo, que o gerente de um distrito municipal precise verificar quais requerentes e beneficiários realmente moram dentro da área atendida pelo distrito. O DB2 Spatial Extender pode derivar estas informações da localização da área atendida e dos endereços de requerentes e destinatários.

Ou suponha que o proprietário de uma cadeia de restaurantes queira fazer negócio em cidades próximas. Para determinar onde abrir novos restaurantes, o proprietário precisa responder a perguntas como: Em que locais destas cidades está concentrada a clientela que geralmente frequenta meus restaurantes? Onde ficam

as principais estradas? Onde é mais baixa a taxa de crimes? Onde se encontram os restaurantes da concorrência? O DB2 Spatial Extender e o DB2 podem gerar informações para responder estas perguntas. Além disso, ferramentas front-end, embora não necessárias, podem participar. Para ilustrar: uma ferramenta de visualização pode colocar informações geradas pelo DB2 Spatial Extender, por exemplo, a localização de concentrações de clientes e a proximidade de rodovias principais a bons restaurantes, no formato de um gráfico em um mapa. As ferramentas de inteligência de negócios podem colocar informações associadas — por exemplo, nomes e descrições de restaurantes concorrentes — em formato de relatório.

Como os dados representam os recursos geográficos

No DB2 Spatial Extender, um recurso geográfico pode ser representado por um ou mais itens de dados; por exemplo, os itens de dados em uma linha de uma tabela. (Um item de dados é o valor ou os valores que ocupam uma célula de uma tabela relacional). Por exemplo, considere edifícios comerciais e residências. Na Figura 1, cada linha da tabela FILIAIS representa uma filial de um banco. De forma semelhante, cada linha da tabela CLIENTES na Figura 1, tomada por inteiro, representa um cliente do banco. No entanto, um subconjunto de cada linha — especificamente os itens de dados que constituem o endereço de um cliente — representam a residência do cliente.

BRANCHES

ID	NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY
937	Airzone-Multern	92467 Airzone Blvd	San Jose	95141	CA	USA

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	95141	CA	USA	A	A

Figura 1. Dados que representam recursos geográficos. A linha de dados na tabela FILIAIS representa uma filial de um banco. Os dados do endereço na tabela CLIENTES representam a residência de um cliente. Os nomes e endereços em ambas as tabelas são fictícios.

As tabelas na Figura 1 contêm dados que identificam e descrevem as filiais e clientes do banco. Esta discussão refere-se a estes dados como dados de negócios.

Um subconjunto de dados de negócios — os valores que indicam os endereços de filiais e de clientes — pode ser convertido em valores a partir dos quais as informações espaciais são geradas. Por exemplo, conforme mostrado na Figura 1, o endereço de uma filial é 92467 Arizone Blvd., San Jose, CA 95141, USA. O endereço do cliente é Rua Concórdia, 9 - São Paulo, SP - CEP 13.951-041 - Brasil. O DB2 Spatial Extender pode converter esses endereços em valores que indicam onde a filial e a residência do cliente estão localizadas, uma em relação à outra. A Figura 2 na página 3 mostra as tabelas FILIAIS e CLIENTES com novas colunas que são designadas para conter tais valores.

BRANCHES

ID	NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	95141	CA	USA	

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	95141	CA	USA		A	A

Figura 2. Tabelas com colunas espaciais incluídas. Em cada tabela, a coluna LOCALIZAÇÃO irá conter as coordenadas que correspondem aos endereços.

Como as informações espaciais serão derivadas dos itens de dados armazenados na coluna LOCALIZAÇÃO, esses itens de dados são referidos nesta discussão como dados espaciais.

A Natureza de Dados Espaciais

Os dados espaciais são compostos por coordenadas que identificam uma localização. O Spatial Extender trabalha com coordenadas bidimensionais especificadas por x e y ou por valores de longitude e latitude.

Uma coordenada é um número que indica:

- Uma posição em um eixo relativa a uma origem, especificada uma unidade de comprimento.
- Uma direção relativa a uma linha de base ou plano, especificada uma unidade de medida angular.

Por exemplo, a latitude é uma coordenada que indica um ângulo relativo ao plano equatorial, geralmente em graus. A longitude é uma coordenada que indica um ângulo relativo ao meridiano de Greenwich, geralmente também em graus. Assim, em um mapa, a posição do Parque Nacional Yellowstone é definida por 44,45 graus de latitude ao norte do equador e por 110,40 graus de longitude a oeste do meridiano de Greenwich. Mais precisamente, essas coordenadas se referem ao centro do Parque Nacional Yellowstone nos EUA.

As definições de latitude e longitude, seus pontos, linhas e planos de referência, unidades de medida e outros parâmetros associados são referidos coletivamente como um sistema de coordenadas. Os sistemas de coordenadas podem ser baseados em valores diferentes da latitude e longitude. Estes sistemas de coordenadas possuem seus próprios pontos, linhas e planos de referência, unidades de medida e parâmetros adicionais associados (como a transformação da projeção).

O item de dados espaciais mais simples consiste em um único par de coordenadas que define a posição de uma única localização geográfica. Um item de dados espaciais mais amplo consiste em várias coordenadas que definem um caminho linear que uma rua ou rio pode formar. Um terceiro tipo consiste em coordenadas que definem o limite de uma área; por exemplo, o limite de um pedaço de terra ou planície aluvial.

Cada item de dados espaciais é uma instância de um tipo de dados espacial. O tipo de dados para coordenadas que marcam uma única localização é ST_Point; o tipo de dados para coordenadas que definem um caminho linear é ST_LineString; e

o tipo de dados para coordenadas que definem o limite de uma área é ST_Polygon. Estes tipos, junto com os outros tipos de dados espaciais, são tipos estruturados que pertencem a uma única hierarquia.

Origem dos Dados Espaciais

Os dados espaciais podem ser obtidos usando-se vários métodos.

Os dados espaciais podem ser:

- Derivados de dados de negócios
- Gerados a partir de funções espaciais
- Importados a partir de origens externas

Utilizando dados de negócios como dados de origem

O DB2 Spatial Extender pode derivar dados espaciais de dados de negócios, como endereços. Esse processo é chamado codificação geográfica.

Para ver a sequência envolvida, considere Figura 2 na página 3 como uma imagem "antes" e Figura 3 como uma imagem "depois". A Figura 2 na página 3 mostra que a tabela BRANCHES e a tabela CUSTOMERS têm uma coluna designada para dados espaciais. Suponha que o DB2 Spatial Extender codifique geograficamente os endereços nessas tabelas para obter coordenadas que correspondam aos endereços e coloque as coordenadas nas colunas. A Figura 3 ilustra esse resultado.

BRANCHES

ID	NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	95141	CA	USA	1653 3094

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	95141	CA	USA	953 1527	A	A

Figura 3. Tabelas que contêm dados espaciais derivados de dados de origem. A coluna LOCATION na tabela CUSTOMERS contém coordenadas derivadas do endereço nas colunas ADDRESS, CITY, POSTAL CODE, STATE_PROV e COUNTRY. De forma semelhante, a coluna LOCATION na tabela BRANCHES contém coordenadas derivadas do endereço nas colunas ADDRESS, CITY, POSTAL CODE, STATE_PROV e COUNTRY desta tabela.

O DB2 Spatial Extender utiliza uma função, chamada de geocodificador, para converter dados de negócios em coordenadas para permitir a operação de funções espaciais nos dados.

Utilizando funções para gerar dados espaciais

Você pode utilizar funções para gerar dados espaciais a partir de dados informados.

Os dados espaciais podem ser gerados não somente por geocodificadores, mas também por outras funções. Suponha, por exemplo, que o banco, cujas filiais estão definidas na tabela FILIAIS, deseja saber quantos clientes estão localizados dentro de cinco milhas de cada filial. Antes que o banco obtenha estas informações do banco de dados, ele precisa definir a região que se encontra em um raio especificado ao redor de cada filial. Uma função do DB2 Spatial Extender, ST_Buffer, pode criar tal definição. Utilizando as coordenadas de cada ramificação

como entrada, o ST_Buffer pode gerar as coordenadas que demarcam os perímetros das regiões. A Figura 4 mostra a tabela FILIAIS com informações fornecidas pelo ST_Buffer.

BRANCHES

ID	NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION	SALES_AREA
937	Airzone-Multern	92467 Airzone Blvd	San Jose	95141	CA	USA	1653 3094	1002 2001, 1192 3564, 2502 3415, 1915 3394, 1002 2001

Figura 4. Tabela que contém novos dados espaciais derivados de dados espaciais existentes. As coordenadas na coluna SALES_AREA foram obtidas pela função ST_Buffer a partir das coordenadas na coluna LOCALIZAÇÃO. Assim como as coordenadas na coluna LOCALIZAÇÃO, as na coluna SALES_AREA são simuladas; elas não são verdadeiras.

Além de ST_Buffer, o DB2 Spatial Extender fornece várias outras funções que derivam novos dados espaciais de dados espaciais existentes.

Importando dados espaciais

O Spatial Extender fornece serviços para importar dados espaciais no formato Shapefile.

Os dados espaciais no formato Shapefile estão disponíveis a partir de muitas origens por meio da Internet. É possível fazer download de dados e mapas dos Estados Unidos e de recursos mundiais, como países, estados, cidades, rios e muitos outros, selecionando a oferta de Dados de Mapa de Amostra do DB2 Spatial Extender na página da Web Versões para Avaliação e Demos disponível em http://www.ibm.com/software/data/spatial/db2spatial.

Você pode importar dados espaciais de arquivos fornecidos por origens de dados externas. Estes arquivos geralmente contêm dados que são empregados em mapas: cruzamentos de ruas, planícies aluviais, deslocamentos por terremotos e outros. Utilizando esses dados junto com dados espaciais gerados por você, é possível aumentar as informações espaciais disponíveis. Se, por exemplo, uma departamento de trabalho público precisa determinar a quais riscos uma comunidade residencial está vulnerável, ele pode usar o ST_Buffer para definir uma região ao redor da comunidade. O departamento de serviço público pode, então, importar dados sobre planícies aluviais e deslocamentos por terremotos para saber quais destas áreas problemáticas abrangem esta região.

Como Recursos, Informações espaciais, Dados espaciais e geometrias são Associados

É fornecido um resumo de vários conceitos básicos que suportam as operações do DB2 Spatial Extender, como recursos geográficos, informações espaciais, dados espaciais e geometrias.

O DB2 Spatial Extender permite obter fatos e figuras relacionados a itens que podem ser definidos geograficamente, ou seja, em termos de sua localização na Terra ou em uma região da Terra. A documentação do DB2 refere-se a tais fatos e figuras como *informações espaciais*, e aos itens como *recursos geográficos* (chamados de *recursos* aqui, para abreviar).

Por exemplo, você pode utilizar o DB2 Spatial Extender para determinar se quaisquer áreas ocupadas sobrepõem o local proposto para um aterro. As áreas

ocupadas e o local proposto são recursos. Um achado, como por exemplo, se existe alguma sobreposição seria um exemplo de informações espaciais. Se for comprovado que existe a sobreposição, a extensão dela também seria um exemplo de informações espaciais.

Para gerar informações espaciais, o DB2 Spatial Extender deve processar dados que definem as localizações dos recursos. Esses dados, denominados *dados espaciais*, consistem em coordenadas que fazem referência às localizações em um mapa ou projeção semelhante. Por exemplo, para determinar se um recurso sobrepõe outro, o DB2 Spatial Extender deve determinar onde as coordenadas de um dos recursos estão situadas com relação às coordenadas do outro.

No mundo da tecnologia da informação espacial, é comum imaginar recursos como sendo representados por símbolos chamados de *geometrias*. As geometrias são parcialmente visuais e parcialmente matemáticas. Considere seu aspecto visual. O símbolo para um recurso que tem largura e extensão, como um parque ou cidade, é uma figura com vários lados. Essa geometria é chamada de *polígono*. O símbolo para um recurso linear, como um rio ou estrada, é uma linha. Essa geometria é chamada de *sequência de linhas*.

Uma geometria tem propriedades que correspondem a fatos sobre o recurso que ela representa. A maioria destas propriedades podem ser expressas matematicamente. Por exemplo, as coordenadas para um recurso constituem coletivamente uma das propriedades da geometria correspondente do recurso. Outra propriedade, chamada *dimension*, é um valor numérico que indica se um recurso possui comprimento ou amplitude.

Dados espaciais e algumas informações espaciais podem ser exibidos em termos de geometrias. Considere o exemplo, descrito anteriormente, das áreas ocupadas e do local proposto para aterro. Os dados espaciais para as áreas ocupadas incluem coordenadas armazenadas em uma coluna de uma tabela em um banco de dados DB2. A convenção deve considerar o que está armazenado não apenas como dados, mas como geometrias reais. Como as áreas ocupadas têm largura e extensão, você pode ver se estas geometrias são polígonos.

Como dados espaciais, algumas informações espaciais também são exibidas em termos de geometrias. Por exemplo, para determinar se uma área ocupada sobrepõe um local proposto para aterro, o DB2 Spatial Extender deve comparar as coordenadas no polígono que representa o local com as coordenadas dos polígonos que representam as áreas ocupadas. As informações resultantes, isto é, as áreas sobrepostas, também são consideradas polígonos: geometrias com coordenadas, dimensões e outras propriedades.

Capítulo 2. Geometrias

No DB2 Spatial Extender, a definição operacional de geometria é “um modelo de recurso geográfico.”

O Webster's Revised Unabridged Dictionary define *geometria* como “A área da matemática que investiga as relações, propriedades e medidas de figuras sólidas, superfícies, linhas e ângulos; a ciência que trata das propriedades e relações de grandezas; a ciência das relações de espaço.” A palavra geometria também é utilizada para indicar as recursos geométricos que, no último milênio ou mais, os cartógrafos vêm utilizando para mapear o mundo. Uma definição abstrata desse novo significado de geometria é “um ponto ou agregado de pontos que representam um recurso no solo”.

No DB2 Spatial Extender, o modelo pode ser expresso em termos das coordenadas do recurso. O modelo contém informações, por exemplo, as coordenadas identificam a posição do recurso em relação a pontos de referência fixos. Além disso, o modelo pode ser utilizado para produzir informações, por exemplo, a função ST_Overlaps pode utilizar as coordenadas de duas regiões próximas como entrada e retornar informações indicando se as regiões estão sobrepostas ou não.

As coordenadas de um recurso que uma geometria representa são consideradas propriedades da geometria. Diversos tipos de geometrias têm outras propriedades também, por exemplo, área, comprimento e limites.

As geometrias às quais o DB2 Spatial Extender oferece suporte formam uma hierarquia, mostrada na figura a seguir. A hierarquia de geometrias é definida pelo documento "OpenGIS Simple Features Specification for SQL" do OGC (OpenGIS Consortium), Inc. Sete membros da hierarquia são instanciáveis. Ou seja, eles podem ser definidos com valores de coordenadas específicos e processados visualmente conforme mostra a figura.

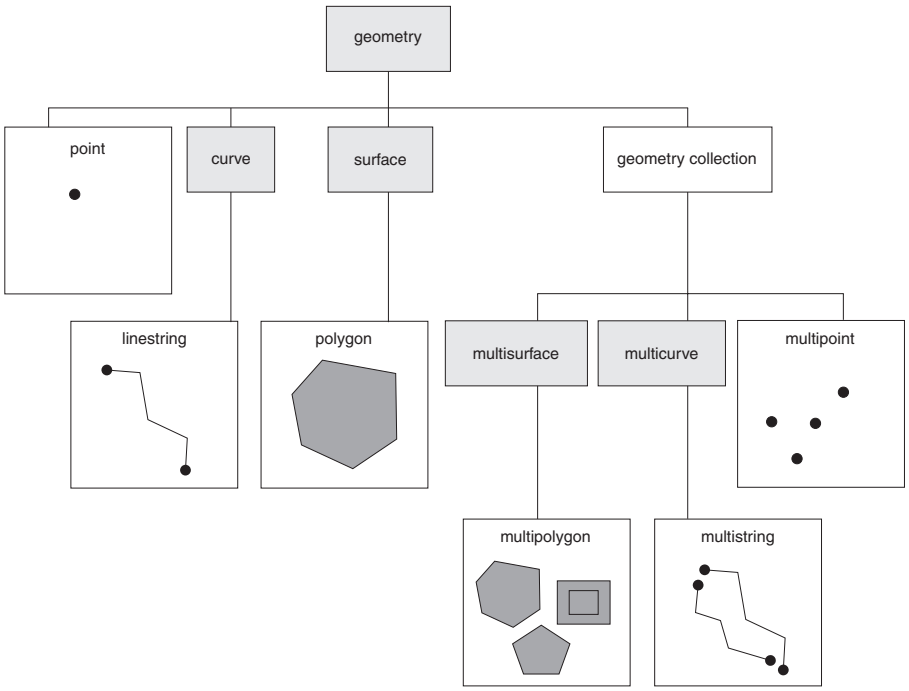


Figura 5. Hierarquia de Geometrias Suportadas pelo DB2 Spatial Extender. As geometrias instanciáveis nesta figura incluem exemplos de como podem ser processadas visualmente.

Os tipos de dados espaciais suportados pelo DB2 Spatial Extender são implementações das geometrias mostradas na figura.

Como a figura indica, uma superclasse chamada geometria é a raiz da hierarquia. O tipo raiz e outros subtipos adequados na hierarquia não são instanciáveis. Além disso, os usuários podem definir seus próprios subtipos adequados, instanciáveis ou não.

Os subtipos são divididos em duas categorias: os subtipos de figura geométrica base e os subtipos de coleção homogênea.

As figuras geométricas base incluem:

Pontos
Um único ponto. Os pontos representam recursos distintos que são notados como ocupantes do local em que uma linha da coordenada leste/oeste (como um paralelo) faz interseção com uma linha da coordenada norte/sul (como um meridiano). Por exemplo, suponha que a notação em um mapa-mundi mostra que cada cidade do mapa está localizada na interseção entre um paralelo e um meridiano. Um ponto pode representar cada cidade.

Sequências de linhas
Uma linha entre dois ou mais pontos. Não precisa ser uma linha reta. As sequências de linhas representam recursos geográficos lineares (por exemplo, ruas, canais e tubulações).

Polígonos
Um polígono ou superfície em um polígono. Polígonos representam recursos geográficos multifacetados (por exemplo, áreas de preservação, florestas e habitats naturais).

As coleções homogêneas incluem:

Multipontos
Uma coleção de geometrias de vários pontos. Multipontos representam recursos de várias partes cujos componentes estão localizados cada um na interseção de uma linha coordenada leste/oeste e uma linha coordenada norte/sul (por exemplo, um arquipélago cujos membros estejam situados cada um em uma interseção de um paralelo e meridiano).

Sequências multilinha
Uma coleção de geometrias de várias curvas com várias sequências de linhas. Sequências multilinha representam recursos de várias partes que são compostos (por exemplo, sistemas fluviais e sistemas rodoviários).

Multipolígonos
Uma coleção de geometrias de várias superfícies com vários polígonos. Multipolígonos representam recursos de várias partes compostos de unidades multifacetadas ou componentes (por exemplo, um grupo de fazendas em uma região específica ou um sistema lacustre).

Como seu nome implica, as coleções homogêneas são coleções de figuras geométricas básicas. Além de compartilhar as propriedades da figura geométrica básica, as coleções homogêneas possuem algumas propriedades próprias também.

Propriedades de Geometrias

Este tópico descreve propriedades que estão disponíveis para geometrias.

Estas propriedades são:
- O tipo ao qual uma geometria pertence
- Coordenadas da geometria
- Um interior, limite e exterior da figura geométrica
- A qualidade de ser simples ou não simples
- A qualidade de ser vazio ou não vazio
- Um retângulo de limite mínimo ou envelope da geometria
- Dimensão
- O identificador do sistema de referência espacial ao qual uma geometria está associada

Tipos de Geometria

Cada geometria pertence a um tipo na hierarquia de geometrias suportadas pelo DB2 Spatial Extender.

Os tipos na hierarquia a serem definidos com valores de coordenadas específicas são:
- Pontos
- Sequências de linhas
- Polígonos
- Coleções de geometrias
- Multipontos
- Sequências multilinha
- Multipolígonos

Coordenadas da geometria

Todas as geometrias incluem no mínimo uma coordenada X e uma Y, a não ser que sejam geometrias vazias. Nesse caso, não conterão coordenadas.

Além disso, uma geometria pode incluir uma ou mais coordenadas Z e coordenadas M. As coordenadas X, Y, Z e M são representadas como números de precisão dupla. Isto é explicado nas seguintes subseções:

- Coordenadas X e Y
- Coordenadas Z
- Coordenadas M

Coordenadas X e Y

Um valor da coordenada X indica um local que é relativo a um ponto de referência a leste ou oeste. Um valor da coordenada Y indica um local que é relativo a um ponto de referência ao norte ou sul.

Coordenadas Z

Para geometrias que possuem uma altitude ou profundidade associada, cada um dos pontos que formam a geometria de um recurso pode incluir uma coordenada Z opcional que representa uma altitude ou profundidade normal para a superfície da terra.

Coordenadas M

Uma coordenada M (medida) é um valor que transmite informações sobre um recurso geográfico e que é armazenada junto com as coordenadas que definem a localização do recurso.

Por exemplo, suponha que você esteja representando estradas em seu aplicativo. Se desejar que seu aplicativo processe valores que indicam distâncias lineares ou marcos divisórios, você pode armazenar estes valores junto com as coordenadas que definem localizações na rodovia. As coordenadas M são representadas como números de precisão dupla.

Interior, Limite e Exterior

Todas as geometrias ocupam uma posição no espaço, definida por seus interiores, limites e exteriores.

O exterior de uma figura geométrica é todo o espaço não ocupado pela figura geométrica. O limite de uma figura geométrica serve como a interface entre seu interior e exterior. O interior é o espaço ocupado pela figura geométrica.

Geometrias Simples ou não Simples

Os valores de alguns subtipos de geometrias (sequências de linhas, multipontos e sequências multilinha) são simples ou não simples. Uma geometria é simples se ela estiver de acordo com todas as regras de topologia impostas ao seu subtipo e não simples se não estiver.

Uma cadeia de linhas é simples se não fizer interseção com seu interior. Um multiponto é simples se nenhum de seus elementos ocupar o mesmo espaço da coordenada. Pontos, superfícies, multisuperfícies e geometrias vazias são sempre simples.

Geometrias Fechadas

Uma curva será fechada se seus pontos inicial e final forem iguais. Uma multicurva será fechada se todos os seus elementos forem fechados. Um anel é uma curva simples, fechada.

Geometrias Vazias ou não Vazias

Uma figura geométrica é vazia se não possuir pontos. O envelope, o limite, o interior e o exterior de uma geometria vazia não estão definidos e serão representados como nulos.

Uma geometria vazia é sempre simples. Os polígonos e multipolígonos vazios têm uma área de valor 0.

MBR (Minimum Bounding Rectangle)

O MBR de uma geometria é a geometria de limite formada pelas coordenadas mínima e máxima (X,Y).

Exceto para os seguintes casos especiais, os MBRs de geometrias formam um retângulo delimitador quando:

- O MBR de qualquer ponto é o próprio ponto, porque suas coordenadas X mínima e máxima são iguais e suas coordenadas Y mínima e máxima são iguais.
- O MBR de uma sequência de linhas horizontal ou vertical é uma sequência de linhas representada pelo limite (os pontos extremos) da sequência de linhas de origem.

Dimensão de Geometrias

Uma geometria pode ter um valor de dimensão para indicar se uma geometria está vazia ou se possui um comprimento ou área.

Os valores de dimensões são os seguintes:

–1 Indica que uma geometria está vazia

0 Indica que uma geometria não possui comprimento e uma área de 0 (zero)

1 Indica que uma geometria possui um comprimento maior que 0 (zero) e uma área de 0 (zero)

2 Indica que uma geometria possui uma área que é maior que 0 (zero)

Os subtipos de ponto e multiponto têm uma dimensão zero. Os pontos representam recursos dimensionais que podem ser modelados com uma única tupla de coordenadas, enquanto os subtipos de multiponto representam dados que devem ser modelados com um conjunto de pontos.

Os subtipos sequência de linhas e multisequências de linhas têm dimensão um. Elas armazenam segmentos de rodovia, extensões de rios e quaisquer outros recursos que sejam lineares na natureza.

Os subtipos polígono e multipolígono possuem uma dimensão de dois. Os recursos cujo perímetro abrange uma área definível, como florestas, terrenos e lagos, podem ser representados pelo tipo de dados polígono ou multipolígonos.

Identificador do sistema de referência espacial

O identificador numérico para um sistema de referência espacial determina qual sistema de referência espacial será utilizado para representar a geometria.

Todos os sistemas de referência espacial conhecidos no banco de dados podem ser acessados por meio da visualização de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Capítulo 3. Como Utilizar o DB2 Spatial Extender

O suporte e o uso do DB2 Spatial Extender envolvem duas atividades principais: configuração do DB2 Spatial Extender e trabalho em projetos que usam dados espaciais. Este tópico apresenta as interfaces que podem ser utilizadas para executar tarefas espaciais.

Interfaces para o DB2 Spatial Extender e Funcionalidade Associada

Há uma grande variedade de interfaces que podem ser usadas para configurar o DB2 Spatial Extender.

É possível usar qualquer uma das interfaces a seguir para configurar o DB2 Spatial Extender e criar projetos que usam dados espaciais:

- Programas de aplicativos que chamam procedimentos armazenados do DB2 Spatial Extender.
- Um CLP (Processador de Linha de Comandos) fornecido pelo DB2 Spatial Extender. O CLP do DB2 Spatial Extender é frequentemente mencionado como o CLP do db2se.
- Consultas SQL que você submete a partir do CLP do DB2 ou de um programa de aplicativo.
- Projetos de software livre que incluem suporte para o DB2 Spatial Extender. Alguns projetos de código aberto com esse suporte são:
  - IBM Data Management Geobrowser para DB2 e Informix é um geobrowser para visualizar tabelas espaciais e os resultados da análise espacial de forma gráfica. Para obter informações adicionais, consulte http://www.ibm.com/developerworks/data/tutorials/dm-1103db2geobrowser/index.html.
  - GeoTools () é uma biblioteca Java para construção de aplicativos espaciais. Para obter informações adicionais, consulte http://www.geotools.org/.
  - GeoServer é um servidor de mapas da Web e servidor de recursos da Web. Para obter informações adicionais, consulte http://geoserver.org/display/GEOS/Welcome.
  - uDIG é um aplicativo de análise e visualização de dados espaciais do desktop. Para obter informações adicionais, consulte http://udig.refractions.net/.
  - Produtos GIS, como Esri ArcGIS

Configurando o DB2 Spatial Extender

Este procedimento descreve as etapas que devem ser executadas para configurar o DB2 Spatial Extender.

Procedimento

Para configurar o DB2 Spatial Extender:

1. Faça planos e preparações (decida quais projetos criar, qual interface ou interfaces utilizar, selecione uma equipe para administrar o DB2 Spatial Extender e criar os projetos, etc.).
2. Instale o DB2 Spatial Extender.

3. Revise suas configurações do parâmetro de configuração do banco de dados e ajuste-as, se necessário. Você deve assegurar que o banco de dados tenha memória e espaço suficientes para funções espaciais, arquivos de log e aplicativos DB2 Spatial Extender.
4. Configure os recursos espaciais para seu banco de dados. Estes recursos incluem um catálogo do sistema, tipos de dados espaciais, funções espaciais, um geocodificador e outros objetos. Para obter informações adicionais, consulte Ativando um banco de dados para operações espaciaisAtivando um banco de dados para operações espaciais.

Exemplo

O exemplo a seguir mostra as etapas executadas por uma empresa fictícia, chamada Safe Harbor Real Estate Insurance Company, para configurar o DB2 Spatial Extender:

1. O ambiente de sistemas de informações da Safe Harbor Real Estate Insurance Company inclui um sistema de banco de dados DB2 e um sistema de arquivos separados apenas para dados espaciais. Até certo ponto, os resultados das consultas podem incluir combinações de dados dos dois sistemas. Por exemplo, uma tabela do DB2 armazena informações sobre rendimentos e um arquivo no sistema de arquivos contém os locais das filiais da empresa. Portanto, é possível saber quais escritórios apresentam rendimentos de quantias especificadas e, então, determinar onde estes escritórios estão localizados. Mas os dados dos dois sistemas não podem ser integrados (por exemplo, os usuários não podem juntar colunas do DB2 com registros do sistema de arquivos e serviços do DB2, como otimização de consultas, não ficam disponíveis para o sistema de arquivos). Para superar estas desvantagens, a Porto Seguro adquire o DB2 Spatial Extender e estabelece um novo departamento de Desenvolvimento Espacial (chamado de departamento Espacial, para abreviar).

   A primeira missão do departamento Espacial é incluir o DB2 Spatial Extender no ambiente DB2 da Porto Seguro:

   - A equipe de gerenciamento do departamento designa uma equipe de administração espacial para instalar e implementar o DB2 Spatial Extender e uma equipe de análise espacial para gerar e analisar informações espaciais.
   - Como a equipe de administração tem profundo conhecimento do UNIX®, ela decide utilizar o CLP do db2se para administrar o DB2 Spatial Extender.
   - Como as decisões de negócios da Porto Seguro são conduzidas principalmente pelas exigências dos clientes, a equipe de gerenciamento decide instalar o DB2 Spatial Extender no banco de dados que contém as informações sobre seus clientes. Grande parte das informações é armazenada em uma tabela chamada CUSTOMERS. A tabela CUSTOMERS tem colunas LATITUDE e LONGITUDE que foram preenchidas como parte do processo de purificação de endereço.
2. A equipe de administração espacial instala o DB2 Spatial Extender em um ambiente DB2 UNIX.
3. Um membro da equipe de administração espacial ajusta as características do log de transações, o tamanho de heap do aplicativo e o tamanho de heap de controle de aplicativo aos valores adequados aos requisitos para o DB2 Spatial Extender.
4. A equipe de administração espacial configura os recursos que serão requeridos pelos projetos que ela está planejando.

- Um membro da equipe emite o comando **db2se enable_db** para obter os recursos que ativam o banco de dados para operações espaciais. Estes recursos incluem o catálogo do DB2 Spatial Extender, tipos de dados espaciais, funções espaciais e outros.

O que Fazer Depois

Depois de configurar o DB2 Spatial Extender, é possível começar a criar projetos que usam dados espaciais.

Criando projetos que utilizam dados espaciais

Depois de configurar o DB2 Spatial Extender, você está pronto para empreender projetos que usem dados espaciais. Este procedimento descreve as etapas envolvidas na criação de tais projetos.

Antes de Iniciar

- Configurar o DB2 Spatial Extender

Procedimento

Para criar um projeto que utiliza dados espaciais:

1. Faça planos e preparações (defina metas para o projeto, decida as tabelas e dados necessários, determine o sistema ou sistemas de coordenadas a serem utilizados e outras coisas).
2. Decida se um sistema de referência espacial existente atende às suas necessidades. Se nenhum atender, crie um.

   Um sistema de referência espacial é um conjunto de valores de parâmetros que inclui:

   - Coordenadas que definem a máxima extensão possível de espaço referido por um determinado intervalo de coordenadas. É necessário determinar o intervalo máximo possível de coordenadas que podem ser determinadas a partir do sistema de coordenadas que está sendo utilizado e selecionar ou criar um sistema de referência espacial que reflita este intervalo.
   - O nome do sistema de coordenadas a partir do qual as coordenadas são derivadas.
   - Números utilizados em operações matemáticas para converter coordenadas recebidas como entrada em valores que podem ser processados com eficiência máxima. As coordenadas são armazenadas em seu formato convertido e retornadas ao usuário em seu formato original.
3. Crie colunas espaciais conforme necessário. Observe que, em muitos casos, se os dados em uma coluna espacial tiverem que ser lidos por uma ferramenta de visualização, a coluna deverá ser a única coluna espacial na tabela ou exibição à qual ela pertence. Como alternativa, se a coluna for uma de várias colunas espaciais em uma tabela, ela poderá ser incluída em uma exibição que não tenha outras colunas espaciais e as ferramentas de visualização poderão ler os dados a partir desta exibição.
4. Configure colunas espaciais para serem acessadas por ferramentas de visualização, conforme necessário, registrando as colunas no catálogo do DB2 Spatial Extender. Quando registrar uma coluna espacial, o DB2 Spatial Extender impõe uma limitação de que todos os dados na coluna devem pertencer ao mesmo sistema de referência espacial. Esta limitação garante a integridade dos dados — uma exigência da maioria das ferramentas de visualização.

5. Preencha as colunas espaciais executando uma das ações a seguir:
   a. Para um projeto que requer que os dados espaciais sejam importados, importe-os.
   b. Para um projeto que requer que uma coluna espacial seja configurada a partir das colunas LATITUDE e LONGITUDE, execute uma instrução SQL UPDATE usando o construtor db2gse.ST_Point.
6. Se necessário, facilite o acesso às colunas espaciais. Isso envolve a definição de índices que permitem que o DB2 acesse dados espaciais rapidamente e a definição de exibições que permitem que os usuários recuperem dados inter-relacionados de forma eficiente. Se deseja que as ferramentas de visualização acessem as colunas espaciais das exibições, será necessário registrar essas colunas no DB2 Spatial Extender também.
7. Analise as informações espaciais geradas e as informações de negócios relacionadas. Esta tarefa envolve a consulta a colunas espaciais e colunas não espaciais relacionadas. Em tais consultas, é possível incluir funções do DB2 Spatial Extender que retornam uma grande variedade de informações. Por exemplo, coordenadas que definem uma zona de segurança proposta em torno de um depósito de lixo tóxico ou a distância mínima entre esse local e a área de casas mais próxima.

Exemplo

O exemplo a seguir é uma continuação do exemplo em Configurando o DB2 Spatial Extender. Ele mostra as etapas executadas pela Safe Harbor Real Estate Insurance Company para criar um projeto para a integração de dados de negócios e espaciais.

1. Um Departamento espacial se prepara para desenvolver um projeto, por exemplo:
   - A equipe de gerenciamento define essas metas para o projeto:
     – Determinar onde estabelecer novas filiais
     – Ajustar prêmios com base na proximidade dos clientes às áreas de risco (áreas com altas taxas de acidentes de tráfego, áreas com altas taxas de criminalidade e zonas de enchentes, terremotos e outros)
   - Este projeto em particular estará relacionado aos clientes e escritórios nos Estados Unidos. Portanto, a equipe de administração espacial decide usar o sistema de coordenadas GCS_NORTH_AMERICAN_1983 para os Estados Unidos fornecido pelo DB2 Spatial Extender.
   - A equipe de administração espacial decide quais os dados necessários para satisfazer as metas do projeto e em que tabelas estes dados ficarão contidos.
2. O DB2 Spatial Extender fornece um sistema de referência espacial, chamado NAD83_SRS_1, que é projetado para ser usado com o GCS_NORTH_AMERICAN_1983. A equipe de administração espacial decide utilizar o NAD83_SRS_1.
3. A equipe de administração espacial define colunas para conter dados espaciais.
   - A equipe verifica se a tabela já contém as colunas LATITUDE e LONGITUDE do cliente. Os valores nessas colunas serão usados depois da conversão para valores de pontos espaciais.
   - A equipe cria as tabelas OFFICE_LOCATIONS e OFFICE_SALES para conter dados armazenados em um sistema de arquivos separado usando um formato de arquivo padrão de mercado. Estes dados incluem os endereços das filiais da Porto Seguro, os dados espaciais originados destes endereços através de um geocodificador, e dados espaciais que definem uma zona

dentro de um raio de cinco milhas ao redor de cada escritório. Os dados derivados do geocodificador serão colocados em uma coluna LOCATION na tabela OFFICE_LOCATIONS e os dados que definem as regiões serão colocados em uma coluna SALES_AREA na tabela OFFICE_SALES.

4. A equipe de administração espacial espera usar as ferramentas de visualização para representar o conteúdo das colunas LOCATION e da coluna SALES_AREA graficamente em um mapa. Portanto, a equipe registra todas as três colunas.
5. A equipe de administração espacial preenche a coluna LOCATION na tabela CUSTOMER, tabela OFFICE_LOCATIONS, tabela OFFICE_SALES e uma nova tabela HAZARD_ZONES:
   - A equipe usa a instrução UPDATE CUSTOMERS SET LOCATION = db2gse.ST_Point(LONGITUDE, LATITUDE,1) para preencher o valor LOCATION de LATITUDE e LONGITUDE.
   - A equipe importa dados para as tabelas OFFICE_LOCATIONS e OFFICE_SALES usando o comando **db2se import_shape**.
   - A equipe cria uma tabela HAZARD_ ZONES, registra suas colunas espaciais e importa dados para ela. Os dados se originam de um arquivo adquirido de um fornecedor de mapas.
6. A equipe de administração espacial cria índices para as colunas registradas. Em seguida, cria uma visualização que junta colunas das tabelas CUSTOMERS e HAZARD ZONES e registra as colunas espaciais nesta visualização.
7. A equipe de análise espacial executa consultas para obter informações que a ajudará a determinar onde estabelecer novas filiais e ajustar prêmios com base na proximidade dos clientes às áreas de risco.

Capítulo 4. Introdução ao DB2 Spatial Extender

Você deve familiarizar-se com as instruções para instalação e configuração do Spatial Extender em sistemas operacionais suportados. Aprenda também como solucionar problemas de instalação e configuração que podem ser encontrados conforme você trabalha com o Spatial Extender.

Configurando e Instalando o DB2 Spatial Extender

A configuração e instalação do DB2 Spatial Extender são necessárias para criar um ambiente que suporta operações espaciais.

Antes de Iniciar

Certifique-se de atender aos requisitos de instalação para produtos de banco de dados DB2 Spatial Extender e DB2. Se o seu sistema não atender aos requisitos de qualquer dos pré-requisitos de software, a instalação falhará.

Sobre Esta Tarefa

Um ambiente do DB2 Spatial Extender consiste em um servidor e em um cliente.

Um servidor DB2 Spatial Extender consiste em uma instalação de servidor de dados DB2 com o software DB2 Spatial Extender instalado.

Um cliente DB2 Spatial Extender consiste em uma instalação do IBM® data server client com o software DB2 Spatial Extender instalado.

Os bancos de dados ativados para operações espaciais estão localizados no servidor. Eles podem ser acessados a partir de um cliente. Os dados espaciais residentes no banco de dados podem ser acessados usando procedimentos armazenados e consultas espaciais do DB2 Spatial Extender do servidor ou do cliente.

Também geralmente parte de um componente típico de um ambiente do DB2 Spatial Extender há um geobrowser. Embora não sejam necessários, eles são úteis para renderizar visualmente os resultados de consultas espaciais, geralmente em forma de mapas. Um geobrowser não está incluído em uma instalação do DB2 Spatial Extender. No entanto, é possível obter o IBM Data Management Geobrowser for DB2 & Informix para visualizar dados espaciais.

Procedimento

Para configurar e instalar o DB2 Spatial Extender em um servidor DB2 ou IBM data server client:

1. Certifique-se de que seu sistema atenda a todos os requisitos do software. Consulte “Requisitos do Sistema para Instalação do Spatial Extender” na página 20.
2. Instale o DB2 Spatial Extender executando uma das seguintes tarefas:
   - “Instalando o DB2 Spatial Extender Usando o Assistente de Configuração do DB2 (Windows)” na página 21

- "Instalando o DB2 Spatial Extender Usando o Assistente de Configuração do DB2 (Linux e UNIX)" na página 23
- Instalando um Produto DB2 Usando um Arquivo de Resposta (Windows)
- Instalando um Produto DB2 Usando um Arquivo de Resposta (Linux e UNIX)

Para instalações usando um arquivo de resposta, você deve indicar as seguintes palavras-chave para a instalação de servidor DB2:

```
INSTALL_TYPE=CUSTOM
COMP=SPATIAL_EXTENDER_CLIENT_SUPPORT
COMP=SPATIAL_EXTENDER_SERVER_SUPPORT
```

Você deve indicar as seguintes palavras-chave para uma instalação do IBM data server client:

```
INSTALL_TYPE=CUSTOM
COMP=SPATIAL_EXTENDER_CLIENT_SUPPORT
```

3. Crie uma instância do DB2 se você ainda não tiver uma. Para fazer isso, use o comando **db2icrt** do DB2 a partir de uma janela de comandos do DB2.
4. Verifique se a instalação do DB2 Spatial Extender foi bem-sucedida testando o ambiente do DB2 Spatial Extender. Consulte "Verificando a instalação do Spatial Extender" na página 24.
5. Opcional: Faça download e instale o IBM Data Management Geobrowser para DB2 e Informix. Uma cópia gratuita pode ser transferida por download de https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-dm-geobrowser.

Informações relacionadas

Analise dados espaciais do DB2 com um geobrowser gratuito

Requisitos do Sistema para Instalação do Spatial Extender

Antes de instalar o DB2 Spatial Extender, certifique-se de que seu sistema atenda a todos os requisitos de espaço de software.

Sistemas operacionais

É possível instalar o DB2 Spatial Extender em sistemas operacionais de 32 bits em computadores baseados em Intel, tais como:

- Windows
- Linux.

Também é possível instalar o DB2 Spatial Extender em sistemas operacionais de 64 bits, tais como:

- AIX,
- HP-UX,
- Solaris Operating System SPARC,
- Linux x86,
- Linux para System z e
- Windows

Requisitos de Software

Para instalar o DB2 Spatial Extender, você deve atender aos requisitos de software para produtos de banco de dados DB2. Para obter detalhes adicionais, consulte "Requisitos de Instalação para Produtos de Banco de Dados DB2" em *Instalando Servidores DB2*.

Spatial Extender em servidores DB2

É possível instalar o Spatial Extender como parte de um produto de banco de dados DB2 ou instalá-lo em uma cópia existente do DB2. Você deve atender aos requisitos de software para os produtos do servidor de banco de dados DB2.

Spatial Extender em IBM data server clients

Se você instalar o IBM data server client em sistemas operacionais Windows, a instalação customizada incluirá o Spatial Extender.

Se você instalar um produto de servidor de banco de dados DB2 em sistemas operacionais AIX, HP-UX, Solaris Operating System, Linux para Intel ou Linux on System z, opcionalmente, o Spatial Extender poderá ser instalado selecionando o tipo de instalação como customizado.

Você deve atender aos requisitos de software para IBM data server client.

Instalando o DB2 Spatial Extender Usando o Assistente de Configuração do DB2 (Windows)

É possível usar o assistente de Configuração do DB2 para instalar o DB2 Spatial Extender em sistemas operacionais Windows como parte de uma instalação do DB2.

Antes de Iniciar

Antes de iniciar o assistente de Configuração do DB2:

- Assegure-se de que o sistema atenda aos requisitos de instalação, memória e disco.
- É necessário possuir uma conta do usuário de Administrador local com os direitos de usuário recomendados para executar a instalação. Em servidores de banco de dados DB2 nos quais LocalSystem pode ser usado como o DAS e o usuário da instância do DB2 e você não estiver usando o recurso de particionamento do banco de dados, um usuário não-administrador com privilégios elevados pode executar a instalação.

  Nota: Se uma conta de usuário não Administrador for executar a instalação do produto, a biblioteca de tempo de execução VS2010 deverá ser instalada antes de tentar instalar um produto de banco de dados DB2. A biblioteca de tempo de execução VS2010 é necessária no sistema operacional antes que o produto de banco de dados DB2 possa ser instalado. A biblioteca de tempo de execução VS2010 está disponível no Web site de download da biblioteca de tempo de execução da Microsoft. Há duas opções: escolha `vcredist_x86.exe` para sistemas de 32 bits ou `vcredist_x64.exe` para sistemas de 64 bits.
- Você deve fechar todos os programas para que o programa de instalação possa atualizar arquivos no computador sem precisar de uma reinicialização.
- Para instalações a partir de uma unidade virtual, você deve mapear a unidade de rede para uma letra da unidade do Windows. O assistente de Configuração

do DB2 não suporta a instalação a partir de uma unidade virtual ou de uma unidade de rede não mapeada (como *hostname**sharename* no Windows Explorer).

Sobre Esta Tarefa

Essa tarefa faz parte da tarefa maior do "Configurando e Instalando o DB2 Spatial Extender" na página 19.

Procedimento

Para instalar o DB2 Spatial Extender como parte de um servidor DB2 ou IBM data server client:

1. Efetue logon no sistema com a conta do Administrador local identificada para a instalação do DB2.
2. Se você tiver o DVD do produto do banco de dados DB2, insira-o na unidade. Se ativado, o recurso de execução automática inicia o Painel de Ativação de Configuração do DB2. Se a execução automática não funcionar, use o Windows Explorer para procurar o DVD do produto de banco de dados DB2 e clique duas vezes no ícone **setup** para iniciar a Barra de Ativação de Configuração do DB2.
3. Se você transferiu por download o produto de banco de dados DB2 a partir do Passport Advantage, execute o arquivo executável para extrair os arquivos de instalação do produto de banco de dados DB2. Use o Windows Explorer para procurar os arquivos de instalação do DB2 e clique duas vezes no ícone **setup** para iniciar a Barra de Ativação do DB2.
4. Revise os pré-requisitos de instalação e as notas sobre a liberação a partir da barra de ativação da Configuração do DB2 para obter informações mais recentes ou é possível prosseguir diretamente para a instalação.
5. Clique em **Instalar um Produto** e a janela Instalar um Produto exibirá os produtos disponíveis para a instalação.
6. Instale o Spatial Extender usando uma das seguintes opções:
   - Para criar uma nova cópia do DB2 com o Spatial Extender, clique em **Instalar Novo** para iniciar a instalação. Selecione o tipo de instalação como customizado para incluir o Spatial Extender como parte da instalação. Continue a instalação seguindo os prompts do assistente de Configuração do DB2.
   - Para instalar o Spatial Extender em uma cópia existente do DB2, clique em **Trabalhar com Existente**. Selecione o tipo de instalação como customizado para incluir o Spatial Extender como parte da instalação. Continue a instalação seguindo os prompts do assistente de Configuração do DB2.

   Após a conclusão da instalação, verifique se há alguma mensagem de aviso ou erro no arquivo de log identificado.

O que Fazer Depois

Depois de instalar o DB2 Spatial Extender, crie uma instância do DB2 nas novas cópias do DB2 e, em seguida, verifique se a instalação foi bem-sucedida.

Instalando o DB2 Spatial Extender Usando o Assistente de Configuração do DB2 (Linux e UNIX)

Para instalar o DB2 Spatial Extender nos sistemas operacionais Linux ou UNIX, use o assistente DB2 Setup ou um arquivo de resposta.

Antes de Iniciar

Antes de iniciar o assistente de Configuração do DB2:

- Assegure-se de que o sistema atenda aos requisitos de instalação, memória e disco.
- Certifique-se de ter um navegador suportado instalado.
- Certifique-se de que a imagem do produto do banco de dados DB2 esteja disponível no computador. É possível obter uma imagem de instalação do DB2 adquirindo um DVD físico do produto de banco de dados DB2 ou fazendo download de uma imagem de instalação a partir do Passport Advantage.
- Se estiver instalando uma versão que não esteja em inglês de um produto do banco de dados DB2, certifique-se de que tenha os Pacotes de Idiomas Nacionais apropriados disponíveis.
- Certifique-se de que tenha instalado o software X Linux capaz de renderizar um usuário gráfico, que o servidor X Linux esteja em execução e a variável *DISPLAY* esteja definida. O assistente Configuração do DB2 é um instalador gráfico.
- Se estiver usando um software de segurança em seu ambiente, você deverá criar usuários necessários do DB2 manualmente antes de iniciar o assistente de Configuração do DB2 .

Sobre Esta Tarefa

Essa tarefa faz parte da tarefa maior do "Configurando e Instalando o DB2 Spatial Extender" na página 19.

É possível instalar um produto do banco de dados DB2 usando a autoridade root ou não root.

Procedimento

Para instalar o DB2 Spatial Extender como parte de um servidor DB2 ou IBM data server client:

1. Se você tiver um DVD físico do produto de banco de dados DB2, vá para o diretório no qual o DVD do produto de banco de dados DB2 está montado, inserindo o seguinte comando:

   ```
   cd /dvdrom
   ```

   onde */dvdrom* representa o ponto de montagem do DVD do produto de banco de dados DB2.
2. Se você tiver transferido por download a imagem do produto de banco de dados DB2, deverá extrair e descompactar arquivo tar do arquivo do produto.
   a. Extraia o arquivo do produto:

      ```
      gzip -d product.tar.gz
      ```

      em que *product* é o nome do produto que foi transferido por download.
   b. Descompacte o arquivo tar do produto:

Em sistemas operacionais Linux

```
tar -xvf product.tar
```

Em sistemas operacionais AIX, HP-UX e Solaris

```
gnutar -xvf product.tar
```

em que *product* é o nome do produto que foi transferido por download.

c. Altere o diretório:

```
cd ./product
```

em que *product* é o nome do produto que foi transferido por download.

Nota: Se você transferiu por download o Pacote de Idiomas Nacionais, descompacte seu arquivo tar no mesmo diretório. Isso irá criar os subdiretórios (por exemplo `./nlpack`) no mesmo diretório e permite que o instalador localize automaticamente as imagens de instalação sem questionamento.

3. Insira o comando **./db2setup** a partir do diretório onde a imagem do produto de banco de dados reside para iniciar o assistente Configuração do DB2.
4. O Painel de Ativação da Configuração do IBM DB2 é aberto. A partir desta janela, você pode visualizar os pré-requisitos de instalação e as notas sobre o release ou pode seguir diretamente para a instalação. Você também pode revisar os pré-requisitos de instalação e as notas sobre o release para obter as informações mais recentes.
5. Clique em **Instalar um Produto** e a janela **Instalar um Produto** exibirá os produtos disponíveis para instalação.
6. Instale o Spatial Extender usando uma das seguintes opções:
   - Para criar uma nova cópia do DB2 com o Spatial Extender, clique em **Instalar Novo** para iniciar a instalação. Selecione o tipo de instalação como customizado para incluir o Spatial Extender como parte da instalação. Continue a instalação seguindo os prompts do assistente de Configuração do DB2.
   - Para instalar o Spatial Extender em uma cópia existente do DB2, clique em **Trabalhar com Existente**. Selecione o tipo de instalação como customizado para incluir o Spatial Extender como parte da instalação. Continue a instalação seguindo os prompts do assistente de Configuração do DB2.

   Após a conclusão da instalação, verifique se há alguma mensagem de aviso ou erro no arquivo de log identificado.

O que Fazer Depois

Depois de instalar o DB2 Spatial Extender, crie uma instância do DB2 nas novas cópias do DB2 e, em seguida, verifique se a instalação foi bem-sucedida.

Verificando a instalação do Spatial Extender

Depois de instalar o DB2 Spatial Extender, você deve validar a instalação.

Antes de Iniciar

Os seguintes pré-requisitos devem ser concluídos antes que você possa validar uma instalação do DB2 Spatial Extender:

- O DB2 Spatial Extender deve ser instalado em um computador.
- Uma instância do DB2 deve ter sido criada no computador em que o servidor de dados do DB2 está instalado.

Sobre Esta Tarefa

Esta tarefa deve ser executada após a tarefa de instalação e configuração do DB2 Spatial Extender. Para obter informações adicionais, consulte "Configurando e Instalando o DB2 Spatial Extender" na página 19. A tarefa inclui a criação de um banco de dados do DB2 e a execução de um aplicativo de amostra do DB2 Spatial Extender fornecido com o DB2 que pode ser usado para verificar se um conjunto central da funcionalidade do DB2 Spatial Extender está funcionando corretamente. Este teste é suficiente para validar se o DB2 Spatial Extender foi instalado e configurado corretamente.

Procedimento

Para verificar a instalação do DB2 Spatial Extender:

1. Linux e UNIX: Efetue logon no sistema com o ID de usuário que corresponde à função do proprietário da instância do DB2.
2. Crie um banco de dados DB2. Para isto, abra uma Janela de Comandos do DB2 e digite o seguinte comando:

   ```
   db2 create database mydb
   ```

   em que *mydb* é o nome do banco de dados.
3. Certifique-se de que você tenha um espaço de tabela temporário do sistema com um tamanho de página de 8 KB ou maior e com um tamanho mínimo de 500 páginas. Se você não tiver esse espaço de tabela, consulte "Criando espaços de tabela temporários" no *Database Administration Concepts and Configuration Reference* para obter detalhes sobre como criá-lo. Esse é um requisito para executar o programa runGSEdemo.
4. Para sistemas operacionais Windows, configure o parâmetro de configuração do gerenciador de banco de dados **agent_stack_sz** como um valor de 100 ou superior. O exemplo a seguir mostra o comando do CLP para configurar o parâmetro como 110:

   ```
   UPDATE DBM CFG USING AGENT_STACK_SZ 110
   ```
5. Vá para o diretório no qual reside o programa runGSEdemo:
   - Para instalações do DB2 Spatial Extender nos sistemas operacionais Linux e UNIX, digite:

     ```
     cd $HOME/sqllib/samples/extenders/spatial
     ```

     em que *$HOME* é o diretório pessoal do proprietário da instância.
   - Para instalação do DB2 Spatial Extender nos sistemas operacionais Windows, digite:

     ```
     cd c:\Arquivos de Programas\IBM\sqllib\samples\extenders\spatial
     ```

     em que *c:\Program Files\IBM\sqllib* é o diretório no qual você instalou o DB2 Spatial Extender.
6. Execute o programa de verificação da instalação. Na linha de comandos do DB2, digite o comando **runGseDemo**:

   ```
   runGseDemo mydb userID password
   ```

   em que *mydb* é o nome do banco de dados.

Capítulo 5. Fazendo Upgrade para o DB2 Spatial Extender Versão 10.1

Fazer Upgrade para o DB2 Spatial Extender para Versão 10.1 em sistemas nos quais você instalou o DB2 Spatial Extender Versão 9.5 ou Versão 9.7 requer mais que instalar o DB2 Spatial Extender para Versão 10.1. Você deve executar a tarefa de upgrade apropriada para esses sistemas.

As tarefas a seguir descrevem todas as etapas para fazer upgrade do DB2 Spatial Extender da Versão 9.5 ou Versão 9.7 para Versão 10.1:

- "Fazendo Upgrade do DB2 Spatial Extender"
- "Atualizando o DB2 Spatial Extender de 32 Bits para 64 Bits" na página 28

Se seu ambiente do DB2 tiver outros componentes, como servidores DB2, clientes e aplicativos de banco de dados, consulte "Fazer Upgrade para o DB2 Versão 10.1" em *Atualizando para DB2 Versão 10.1* para obter detalhes sobre como fazer upgrade desses componentes.

Fazendo Upgrade do DB2 Spatial Extender

O upgrade do DB2 Spatial Extender requer primeiro o upgrade do servidor DB2 e, em seguida, o upgrade de objetos de banco de dados e de dados específicos em bancos de dados espaciais ativados.

Antes de Iniciar

Antes de iniciar o processo de upgrade:

- Certifique-se de que seu sistema atenda aos requisitos de instalação para o DB2 Spatial Extender Versão 10.1.
- Certifique-se de que tenha as autoridades DBADM e DATAACCESS nos bancos de dados espaciais ativados.
- Certifique-se de que você tenha um espaço de tabela temporário do sistema com um tamanho de página de 8 KB ou maior e com um tamanho mínimo de 500 páginas.

Sobre Esta Tarefa

Se você instalou o DB2 Spatial Extender Versão 9.5 ou Versão 9.7, deverá concluir as seguintes etapas antes de usar um banco de dados espacial ativado existente com o DB2 Spatial Extender Versão 10.1. Este tópico descreve as etapas necessárias para upgrade de bancos de dados espaciais ativados de uma versão anterior do DB2 Spatial Extender.

Procedimento

Para fazer upgrade do DB2 Spatial Extender para o Versão 10.1:

1. Faça upgrade do servidor DB2 da Versão 9.5 ou Versão 9.7 para o Versão 10.1 usando uma das tarefas a seguir:
   - "Fazendo upgrade de servidores DB2 (Windows)" no *Atualizando para DB2 Versão 10.1*

- "Fazendo upgrade de servidores DB2 (Linux e UNIX)" no *Atualizando para DB2 Versão 10.1*

Na tarefa de upgrade, você deve instalar o DB2 Spatial Extender Versão 10.1 depois de instalar o DB2 Versão 10.1 para que o upgrade de suas instâncias seja bem-sucedido.

2. Encerre todas as conexões com o banco de dados.
3. Faça upgrade de bancos de dados espaciais ativados da Versão 9.5 ou da Versão 9.7 para o Versão 10.1 usando o comando **db2se upgrade**.

Resultados

Verifique o arquivo de mensagens para obter detalhes sobre os erros recebidos. O arquivo de mensagens também contém informações úteis como índices, visualizações e a configuração de geocodificação da qual foi feito o upgrade.

Atualizando o DB2 Spatial Extender de 32 Bits para 64 Bits

Fazer a atualização do DB2 Spatial Extender Versão 10.1 de 32 bits para o DB2 Spatial Extender Versão 10.1 de 64 bits requer que você faça backup dos índices espaciais, atualize o servidor DB2 para DB2 Versão 10.1 de 64 bits, instale o DB2 Spatial Extender Versão 10.1 de 64 bits e restaure os índices espaciais dos quais você fez backup.

Antes de Iniciar

- Certifique-se de que você tenha um espaço de tabela temporário do sistema com um tamanho de página de 8 KB ou maior e com um tamanho mínimo de 500 páginas.

Sobre Esta Tarefa

Se estiver fazendo upgrade do DB2 Spatial Extender Versão 9.5 ou Versão 9.7 de 32 bits para o DB2 Spatial Extender Versão 9.7 de 64 bits, será necessário executar a tarefa "Fazendo Upgrade do DB2 Spatial Extender" na página 27.

Procedimento

Para atualizar o servidor DB2 Spatial Extender Versão 9.7 de 32 bits para o DB2 Spatial Extender Versão 9.7 de 64 bits:

1. Faça backup de seu banco de dados.
2. Salve os índices espaciais existentes emitindo o comando **db2se save_indexes** a partir de um prompt de comandos do sistema operacional.
3. Atualize o servidor DB2 de 32 bits da Versão 9.5 ou Versão 9.7 para o DB2 Versão 10.1 de 64 bits usando uma das tarefas a seguir:
   - "Atualizando as Instâncias do DB2 de 32 bits para Instâncias de 64 bits (Windows)" em *Instalando Servidores DB2* .
   - "Atualizando cópias do DB2 (Linux e UNIX)" no *Database Administration Concepts and Configuration Reference*.
4. Instale o DB2 Spatial Extender Versão 10.1.
5. Restaure os índices espaciais emitindo o comando **db2se restore_indexes** a partir de um prompt de comandos do sistema operacional.

Capítulo 6. Configurando recursos espaciais para um banco de dados

Depois de configurar o banco de dados para adaptar os dados espaciais, você está pronto para fornecer ao banco de dados os recursos que serão necessários ao criar e gerenciar colunas espaciais e analisar dados espaciais.

Os recursos espaciais incluem:

- Objetos fornecidos pelo Spatial Extender para suportar operações espaciais. Por exemplo, procedimentos armazenados para administrar um banco de dados e tipos de dados espaciais e utilitários para codificação geográfica, importação ou exportação de dados espaciais.
- Geocodificadores fornecidos por usuários ou fornecedores.

Para disponibilizar estes recursos, você deve ativar o banco de dados para operações espaciais, configurar o acesso a dados de referência e registrar geocodificadores de terceiros.

Inventário de Recursos fornecidos para o Banco de Dados

O DB2 Spatial Extender fornece ao banco de dados diversos recursos para ativá-lo para suporte de operações espaciais.

Estes recursos são:

- Procedimentos armazenados. Ao solicitar uma operação espacial — por exemplo, quando emitir um comando para importar dados espaciais — o DB2 Spatial Extender chamará um destes procedimentos armazenados para executar a operação.
- Tipos de dados espaciais. Você deve atribuir um tipo de dados espaciais a cada coluna da tabela ou exibição que deve conter os dados espaciais.
- Catálogo do DB2 Spatial Extender. Algumas operações dependem deste catálogo. Por exemplo, antes de acessar uma coluna espacial a partir das ferramentas de visualização, a ferramenta pode requerer que a coluna espacial esteja registrada no catálogo.
- Um índice de grade espacial. Permite definir índices de grade em colunas espaciais.
- Funções espaciais. Você as utiliza para trabalhar com dados espaciais de diversas formas; por exemplo, para determinar relacionamentos entre geometrias e para gerar mais dados espaciais.
- Definições de sistemas de coordenadas.
- Sistemas de referência espacial padrão.
- Dois esquemas: DB2GSE e ST_INFORMTN_SCHEMA. O DB2GSE contém os objetos que estão apenas listados: procedimentos armazenados, tipos de dados espaciais, o catálogo do DB2 Spatial Extender e outros. As exibições do catálogo também estão disponíveis no ST_INFORMTN_SCHEMA para estar de acordo com o padrão SQL/MM.

Ativando um Banco de Dados para Operações espaciais

A ativação do banco de dados para operações espaciais consiste em fazer com que o DB2 Spatial Extender forneça um banco de dados com recursos para a criação de colunas espaciais e em manipular dados espaciais.

Antes de Iniciar

Antes de ativar um banco de dados para operações espaciais:

- Certifique-se de que o ID do usuário tenha autoridade DBADM no banco de dados.
- Certifique-se de que você tenha um espaço de tabela temporário do sistema com um tamanho de página de 8 KB ou maior e com um tamanho mínimo de 500 páginas.

Procedimento

Ative um banco de dados para operações espaciais de qualquer uma das maneiras a seguir:

- Emita o comando **db2se enable_db**.
- Execute um aplicativo que chama o procedimento DB2GSE.ST_ENABLE_DB.

Você pode escolher explicitamente o espaço de tabelas no qual deseja que o catálogo do DB2 Spatial Extender resida. Se você não fizer isso, o sistema de banco de dados DB2 usará o espaço de tabela padrão.
Se você deseja usar dados espaciais em um ambiente de banco de dados particionado, não use o espaço de tabela padrão. Para obter os melhores resultados, ative o banco de dados em um espaço de tabela definido para um único nó. Geralmente, um único espaço de tabela de nó é definido para tabelas pequenas nas quais o particionamento não é útil.
Por exemplo, é possível definir um único espaço de tabela de nó usando o processador de linha de comandos db2se:

```
db2se enable_db my_db -tableCreationParameters "IN NODE0TBS"
```

Muitas consultas podem ser executadas mais eficientemente se a tabela de sistemas de referência espaciaisDB2GSE.GSE_SPATIAL_REFERENCE_SYSTEMS for replicada ao longo de todos os nós que são usados para tabelas de negócios que serão consultadas. É possível recriar a tabela DB2GSE.GSE_SRS_REPLICATED_AST com instruções como essa:

```
drop table db2gse.gse_srs_replicated_ast;
-- MQT para replicar as informações de SRS entre todos os nós
-- em um ambiente de banco de dados particionado
CREATE TABLE db2gse.gse_srs_replicated_ast AS
   ( SELECT srs_name, srs_id, x_offset, x_scale, y_offset, z_offset, z_scale,
       m_offset, m_scale, definition
     FROM db2gse.gse_spatial_reference_systems )
   DATA INITIALLY DEFERRED
   REFRESH IMMEDIATE
   ENABLE QUERY OPTIMIZATION
   REPLICATED
   IN ts_data  partitioned_tablespace
;

REFRESH TABLE db2gse.gse_srs_replicated_ast;

CREATE  INDEX db2gse.gse_srs_id_ast
   ON db2gse.gse_srs_replicated_ast ( srs_id )
```

```
;

RUNSTATS ON TABLE db2gse.gse_srs_replicated_ast and indexes all;
```

Registering a geocoder

Antes de poder ser usados, os geocodificadores devem ser registrados.

Antes de Iniciar

Antes de registrar um geocodificador, seu ID do usuário deve conter autoridade DBADM no banco de dados no qual o geocodificador reside.

Procedimento

Você pode registrar um geocodificador de uma das seguintes formas:

- Emita o comando **db2se register_gc**.
- Execute um aplicativo que chama o procedimento DB2GSE.ST_REGISTER_GEOCODER.

Capítulo 7. Configurando recursos espaciais para um projeto

Depois que o banco de dados é ativado para operações espaciais, você está pronto para criar projetos que utilizam dados espaciais.

Entre os recursos requeridos para cada projeto estão um sistema de coordenadas seguido pelos dados e um sistema de referência espacial que define a extensão da área geográfica que é referenciada pelos dados.

É importante aprender sobre a natureza de sistemas de coordenadas e o que são sistemas de referência espacial.

Como Utilizar Sistemas de Coordenadas

Ao planejar um projeto que utiliza dados espaciais, você precisa determinar se os dados devem ser baseados em um dos sistemas de coordenadas que são registrados no catálogo do Spatial Extender.

Se nenhum desses sistemas de coordenadas atender aos requisitos, você poderá criar um que atenda aos requisitos. Esta discussão explica o conceito de sistemas de coordenadas e apresenta as tarefas para selecionar um para ser utilizado e criar um novo.

Sistemas de Coordenadas

Um sistema de coordenadas é uma estrutura para definir as localizações relativas de elementos em uma determinada área; por exemplo, uma área da superfície terrestre ou a superfície terrestre como um todo.

O DB2 Spatial Extender suporta os seguintes tipos de sistemas de coordenadas para determinar o local de um recurso geográfico:

Sistema de Coordenadas Geográficas
Um sistema de *coordenadas geográficas* é aquele sistema de referências que utiliza uma superfície esférica tridimensional para determinar localizações na Terra. Qualquer localização na Terra pode ser referida por um ponto com coordenadas de latitude e longitude baseadas em unidades de medida angulares.

Sistema de Coordenadas Projetadas
Um *sistema de coordenadas projetadas* é uma representação plana, bidimensional da Terra. Ele utiliza coordenadas retilíneas (Cartesianas) baseadas em unidades de medida lineares. É baseado em um modelo terrestre esférico (ou esferoidal) e suas coordenadas estão relacionadas a coordenadas geográficas por uma transformação de projeção.

Sistema de Coordenadas Geográficas

Um *sistema de coordenadas geográficas* é aquele que utiliza uma superfície esférica tridimensional para determinar localizações na Terra. Qualquer localização na Terra pode ser referida por um ponto com coordenadas de longitude e latitude.

Por exemplo, a Figura 6 na página 34 mostra um sistema de coordenadas geográficas no qual uma localização é representada pelas coordenadas de 80 graus

de longitude ao Leste e 55 graus de latitude ao Norte.

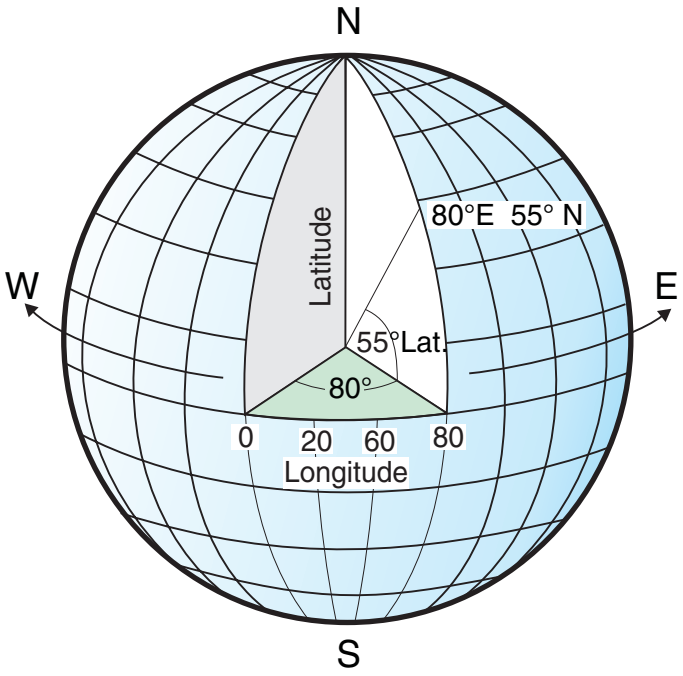


Figura 6. Um sistema de coordenadas geográficas

Cada uma das linhas que percorrem o leste e o oeste possui um valor de latitude constante e é chamada de *paralelas*. Elas são equidistantes e paralelas e formam círculos concêntricos ao redor da Terra. O *equador* é o maior círculo e divide a Terra ao meio. Tem a mesma distância a partir de cada pólo e o valor desta linha latitudinal é zero. As localizações ao norte do equador possuem latitudes positivas que vão de 0 a +90 graus, enquanto as localizações ao sul do equador possuem latitudes negativas que vão de 0 a -90 graus.

A Figura 7 na página 35 ilustra linhas latitudinais.

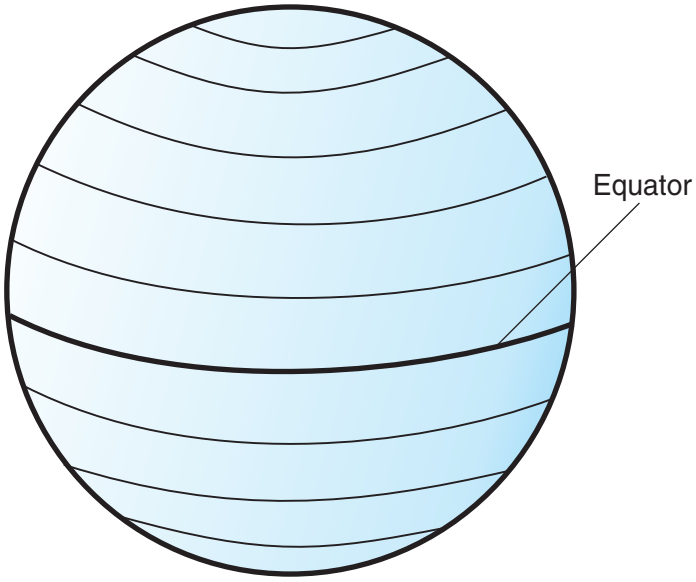


Figura 7. Linhas latitudinais

Cada uma das linhas que percorrem o norte e o sul possui um valor de longitude constante e é chamada de *meridianas*. Elas formam círculos do mesmo tamanho ao redor da terra e se cruzam nos pólos. O *primeiro meridiano* é a linha longitudinal que define a origem (zero grau) para coordenadas longitudinais. Uma das localizações mais utilizadas da linha do meridiano é a linha que passa por Greenwich, Inglaterra. No entanto, outras linhas longitudinais, como as que passam por Berna, Bogotá e Paris, também foram utilizadas como o meridiano principal. As localizações ao leste do primeiro meridiano até seu meridiano *antipodal* (a continuação do primeiro meridiano no outro lado do globo) possuem longitudes positivas que vão de 0 a +180 graus. As localizações a oeste do meridiano principal possuem longitudes negativas que vão de 0 a –180 graus.

A Figura 8 ilustra linhas longitudinais.

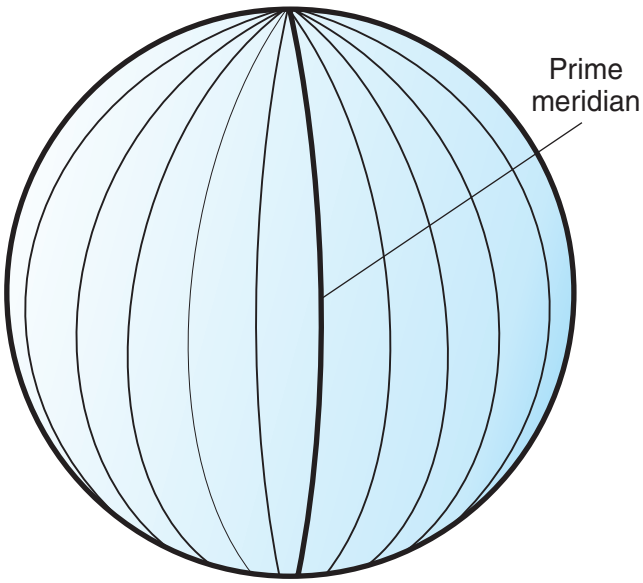


Figura 8. Linhas longitudinais

As linhas latitudinais e longitudinais podem cobrir o globo para formar uma grade, chamada de *gratícula*. O ponto de origem da gratícula é (0,0), em que a linha do equador e a linha do meridiano se cruzam. O equador é o único lugar na gratícula em que a distância linear correspondente à latitude de um grau é aproximadamente igual à distância correspondente à longitude de um grau. Como as linhas longitudinais convergem nos pólos, a distância entre dois meridianos é diferente em cada paralelo. Portanto, conforme você se aproxima dos pólos, a distância correspondente à latitude de um grau será muito maior do que a correspondente à longitude de um grau.

Também é difícil determinar as profundidades das linhas latitudinais utilizando a gratícula. As linhas latitudinais são círculos concêntricos que ficam menores próximos aos pólos. Elas formam um único ponto nos pólos nos quais os meridianos começam. No equador, um grau de longitude equivale a aproximadamente 111.321 quilômetros, enquanto a 60 graus de latitude, um grau de longitude equivale a apenas 55.802 km (esta aproximação é baseada no esferóide Clarke 1866). Portanto, como não existe nenhum comprimento uniforme de graus de latitude e longitude, a distância entre pontos não pode ser calculada com precisão utilizando unidades de medida angulares.

A Figura 9 mostra as diferentes dimensões entre localizações na gratícula.

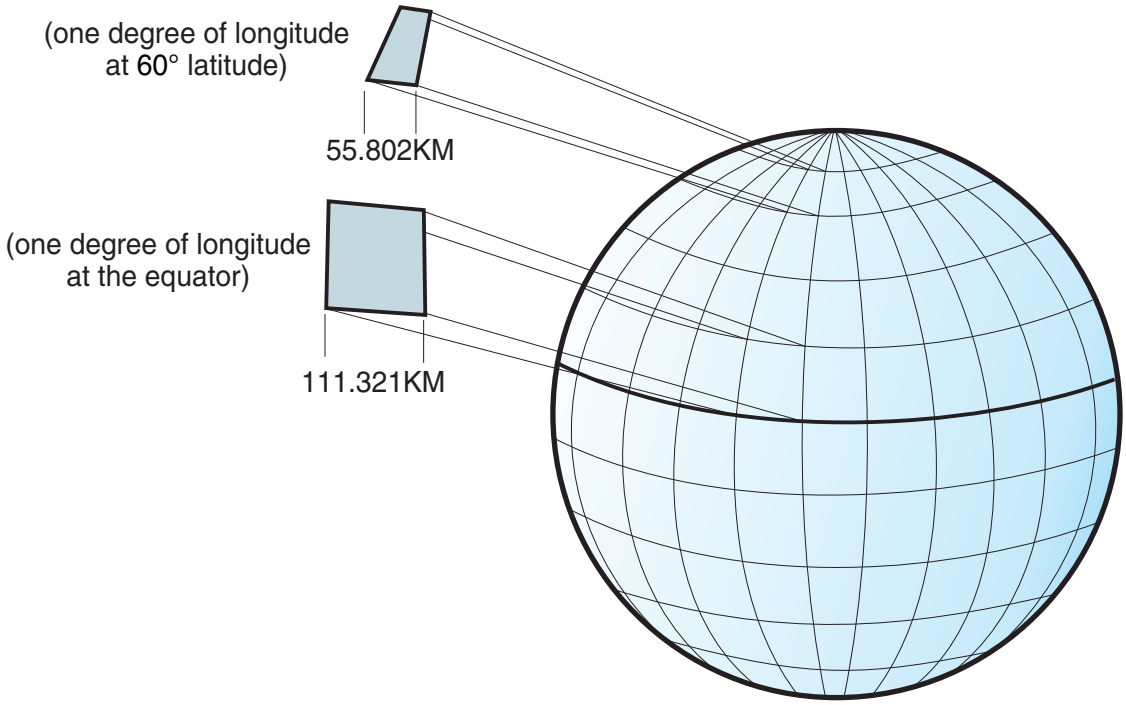


Figura 9. Diferentes dimensões entre localizações na gratícula

Um sistema de coordenadas pode ser definido por uma aproximação de esfera ou esferóide do formato da Terra. Como a Terra não é perfeitamente redonda, um esferóide pode ajudar a manter a precisão de um mapa, dependendo da localização na Terra. Um *esferóide* é um elipsóide baseado em uma elipse, enquanto uma esfera é baseada em um círculo.

O formato da elipse é determinado por dois raios. O raio mais longo é chamado de semi-eixo maior e o raio mais curto é chamado de semi-eixo menor. Um elipsóide é um formato tridimensional formado pela rotação de uma elipse em torno de um de seus eixos.

A Figura 10 mostra as aproximações de esfera e de esferóide da Terra e os eixos maior e menor de uma elipse.

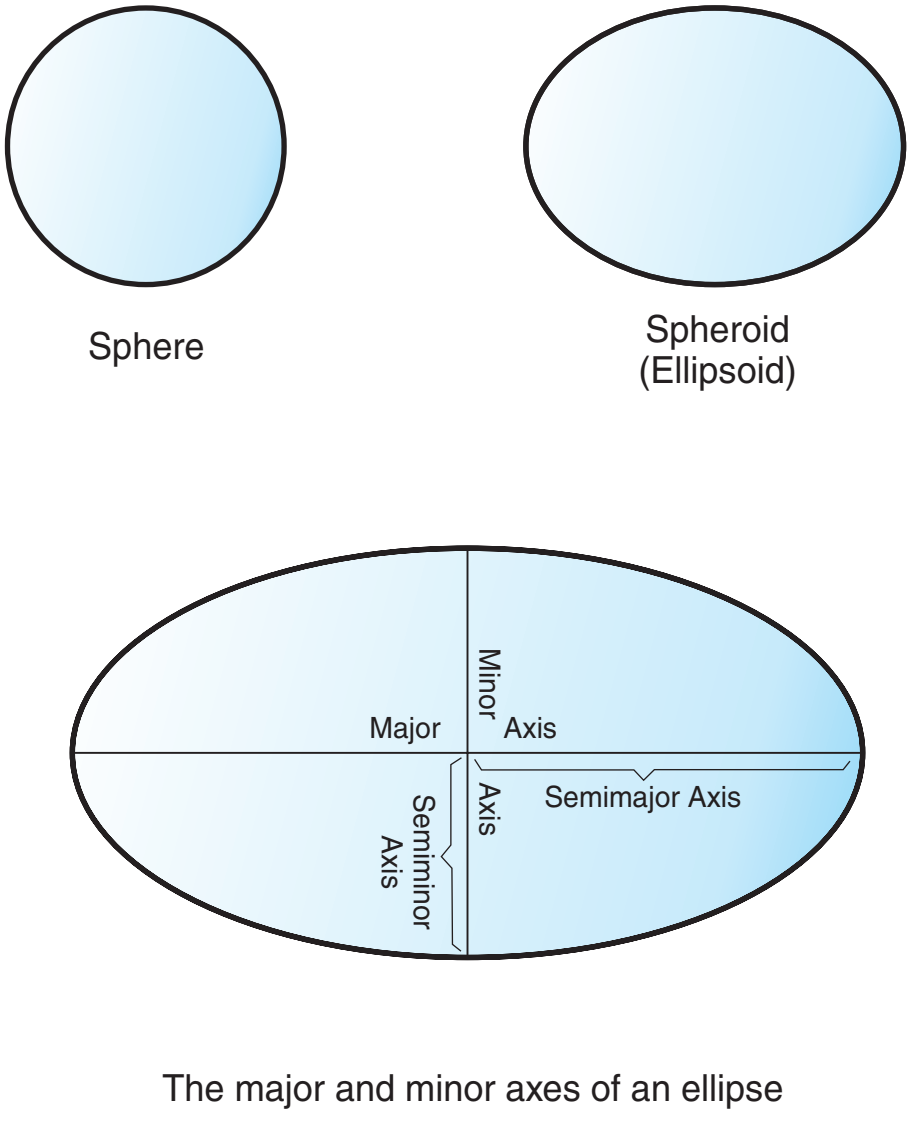


Figura 10. Aproximações de esfera e de esferóide

Um *datum* é um conjunto de valores que definem a posição do esferóide relativo ao centro da Terra. O datum fornece um quadro de referência para medir localizações e define a origem e a orientação de linhas latitudinais e longitudinais. Alguns datums são globais e destinam-se a fornecer uma boa média de precisão ao redor do mundo. Um datum local alinha seu esferóide para ajustar a superfície da Terra em uma determinada área. Portanto, as medidas do sistema de coordenadas não serão precisas se forem utilizadas com uma área diferente da área para a qual foram designadas.

A Figura 11 mostra as diferenças de alinhamento de datums com a superfície da Terra. O datum local, NAD27, está mais alinhado com a superfície da Terra do que o datum centralizado na Terra, WGS84, nesta localização específica.

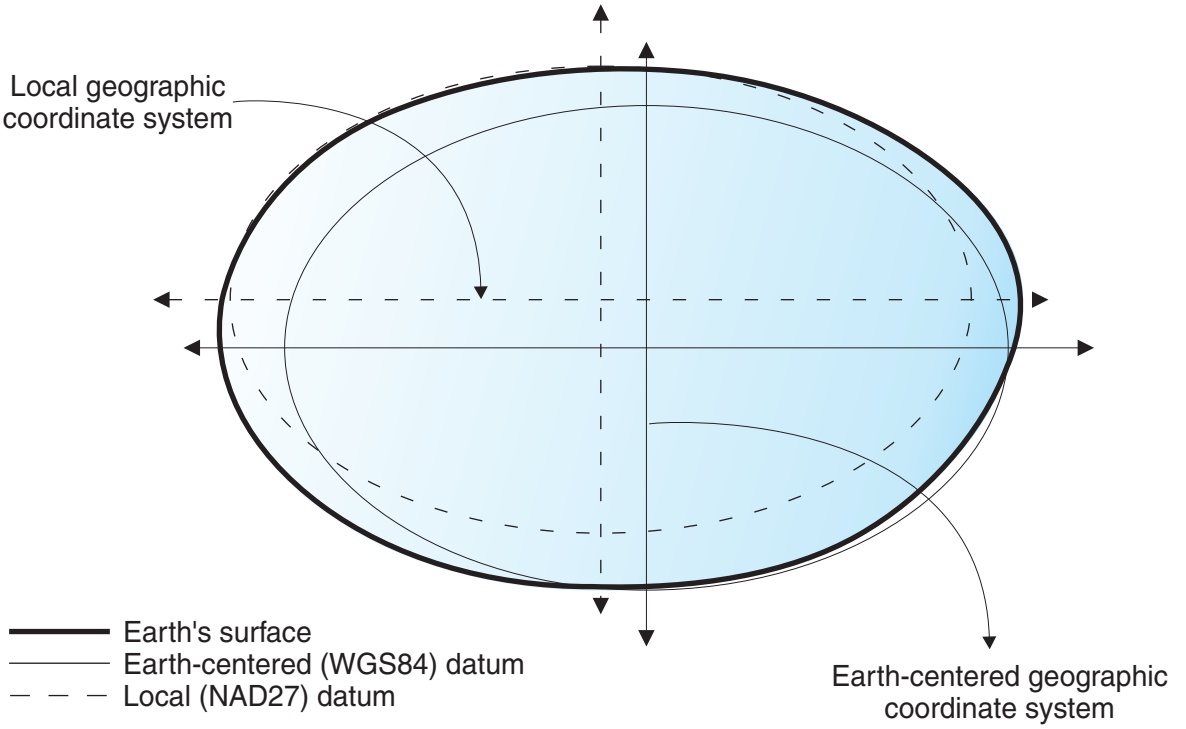


Figura 11. Alinhamentos de datums

Sempre que você alterar o dado, o sistema de coordenadas geográficas é alterado e os valores de coordenadas serão alterados. Por exemplo, as coordenadas no DMS de um ponto de controle em Redlands, Califórnia utilizando o North American Datum de 1983 (NAD 1983) são: "-117 12 57.75961 34 01 43.77884" As coordenadas do mesmo ponto no North American Datum de 1927 (NAD 1927) são: "-117 12 54.61539 34 01 43.72995".

Sistemas de Coordenadas Projetadas

Um *sistema de coordenadas projetadas* é uma representação plana, bidimensional da Terra. Ele é baseado em um sistema de coordenadas geográficas de esfera ou esferóide, mas utiliza unidades de medida lineares para coordenadas, para que os cálculos de distância e área sejam feitos facilmente em relação a estas mesmas unidades.

As coordenadas latitudinais e longitudinais são convertidas em coordenadas x, y na projeção plana. A coordenada x geralmente representa a direção leste de um ponto e a coordenada y geralmente representa a direção norte de um ponto. A linha central que percorre o leste e o oeste é referida como o eixo de x e a linha central que percorre o norte e o sul é referida como o eixo de y.

A interseção dos eixos de x e de y é a origem e geralmente possui uma coordenada (0,0). Os valores acima do eixo x são positivos e os valores abaixo do eixo x são negativos. As linhas paralelas ao eixo de x são equidistantes umas das outras. Os valores à direita do eixo y são positivos e os valores à esquerda do eixo y são negativos. As linhas paralelas ao eixo de y são equidistantes.

São utilizadas fórmulas matemáticas para converter um sistema de coordenadas geográficas tridimensional em um sistema de coordenadas projetado plano bidimensional. A transformação é referida como uma *projeção de mapa*. As projeções de mapas, geralmente, são classificadas pela superfície da projeção utilizada, como superfícies cônicas, cilíndricas e planas. Dependendo da projeção utilizada, as propriedades espaciais diferentes aparecerão distorcidas. As projeções são designadas para reduzir a distorção de uma ou duas características de dados, ainda que a distância, área, forma, direção ou uma combinação destas propriedades podem não ser representações precisas dos dados que estão sendo modelados. Existem vários tipos de projeções disponíveis. Enquanto a maioria das projeções de mapas tentam preservar alguma precisão das propriedades espaciais, existem outras que tentam reduzir a distorção geral, como a projeção de *Robinson*. Os tipos mais comuns de projeções de mapa incluem:

Projeções de áreas iguais
Estas projeções preservam a área de recursos específicos. Estas projeções distorcem a forma, o ângulo e a escala. A projeção *Albers Equal Area Conic* é um exemplo de uma projeção de áreas iguais.

Projeções conformais
Estas projeções preservam a forma local para pequenas áreas. Estas projeções preservam ângulos individuais para descrever relacionamentos espaciais mostrando linhas de gratículas perpendiculares que se cruzam em ângulos de 90 graus no mapa. Todos os ângulos são preservados; porém, a área do mapa é distorcida. As projeções *Mercator* e *Lambert Conformal Conic* são exemplos de projeções conformais.

Projeções equidistantes
Estas projeções preservam as distâncias entre determinados pontos, mantendo a escala de um determinado conjunto de dados. Algumas das distâncias podem ser distâncias reais, que são as mesmas distâncias na mesma escala do globo. Se você sair do conjunto de dados, a escala ficará mais distorcida. A projeção *Sinuosa* e a projeção *Cônica equidistante* são exemplos de projeções equidistantes.

Projeções de direção real ou azimutais
Estas projeções preservam a direção de um ponto para todos os demais pontos, mantendo alguns dos arcos de círculos grandes. Estas projeções fornecem as direções ou ângulos de fundo de todos os pontos no mapa corretamente em relação ao centro. Os mapas azimutais podem ser combinados com projeções de áreas iguais, conformais e equidistantes. A projeção *Lambert Equal Area Azimutal* e a projeção *Azimutal Equidistante* são exemplos de projeções azimutais.

Determinar Qual Sistema de Coordenadas Usar

A primeira etapa no planejamento de um projeto é determinar qual sistema de coordenadas utilizar.

Antes de Iniciar

Antes de criar um sistema de coordenadas, seu ID do usuário deve ter a autoridade DBADM no banco de dados que foi ativado para operações espaciais. Não é necessária nenhuma autorização para utilizar um sistema de coordenadas existente.

Sobre Esta Tarefa

Depois de ativar um banco de dados para operações espaciais, você estará pronto para planejar projetos que utilizem dados espaciais. Você pode utilizar um sistema de coordenadas que foi fornecido com o DB2 Spatial Extender ou um que foi criado por outra pessoa. Mais de 2000 sistemas de coordenadas são fornecidos com o DB2 Spatial Extender.

Para obter mais informações sobre esses sistemas de coordenadas e para determinar quais outros sistemas de coordenadas foram fornecidos com o DB2 Spatial Extender e quais sistemas de coordenadas (se houver algum) foram criados por outros usuários, consulte a visualização do catálogo DB2SE.ST_COORDINATE_SYSTEMS.

Procedimento

Para determinar qual sistema de coordenadas usar:

1. Revise os sistemas de coordenadas existentes que são enviados com o DB2 Spatial Extender e use o sistema de referência espacial correspondente que é baseado no sistema de coordenadas de sua escolha.

   Os sistemas de coordenadas mais usados são:

   - GCS_NORTH_AMERICAN_1983. Utilize este sistema de coordenadas quando precisar definir localizações nos Estados Unidos; por exemplo:
     - Quando você importa dados espaciais para os Estados Unidos a partir dos Dados do Mapa Espacial disponíveis para download.
   - Um sistema de coordenadas ao qual o DB2 Spatial Extender se refere como Não especificado. Utilize este sistema de coordenadas quando:
     - Precisar definir localizações que não têm relacionamento direto com a superfície terrestre; por exemplo, localizações de escritórios em um edifício comercial ou localizações de prateleiras em um armazém.
     - Você pode definir estas localizações em termos de coordenadas positivas que incluem poucos ou nenhum valor decimal.
2. Se não for possível localizar um sistema de coordenadas existente no Spatial Extender, será possível criar um usando um dos métodos a seguir:
   - Emita o comando db2se create_cs a partir do processador da linha de comandos do db2se.
   - Execute um aplicativo que chama o procedimento DB2SE.ST_CREATE_COORDSYS.

   Criar um sistema de coordenadas é raramente necessário. Você também precisa criar um sistema de referência espacial baseado no novo sistema de coordenadas.

Como Configurar Sistemas de Referência Espacial

Ao planejar um projeto que utiliza dados espaciais, você precisa determinar se qualquer um dos sistemas de referência espacial disponíveis pode ser utilizado para esses dados.

Se nenhum dos sistemas disponíveis for apropriado para os dados, você poderá criar um apropriado. Esta seção explica o conceito de sistemas de referência espacial e descreve as tarefas para selecionar um para ser utilizado e criar um novo.

Sistemas de Referência Espacial

Um *sistema de referência espacial* é um conjunto de parâmetros usado para representar uma geometria.

Estes parâmetros são:

- O nome do sistema de coordenadas a partir do qual as coordenadas são derivadas.
- O identificador numérico que identifica exclusivamente o sistema de referência espacial.
- Coordenadas que definem a máxima extensão possível de espaço referido por um determinado intervalo de coordenadas.
- Números que, quando aplicados em certas operações matemáticas, convertem coordenadas recebidas como entrada em valores que podem ser processados com eficiência máxima.

As seções a seguir discutem os valores de parâmetros que definem um identificador, uma extensão máxima de espaço e fatores de conversão.

Identificador do sistema de referência espacial

O SRID (spatial reference system identifier, identificador do sistema de referência espacial) é utilizado como um parâmetro de entrada para diversas funções espaciais.

Definindo o espaço que abrange coordenadas armazenadas em uma coluna espacial

As coordenadas em uma coluna espacial geralmente definem localizações que se estendem por parte da Terra. O espaço no qual ocorre a extensão — de leste a oeste e de norte a sul — é chamado de *extensão espacial*. Por exemplo, considere um grupo de planícies aluviais cujas coordenadas estão armazenadas em uma coluna espacial. Suponha que mais a oeste e mais a leste destas coordenadas estejam valores latitudinais de –24.556 e –19.338, respectivamente, e que mais ao norte e mais ao sul das coordenadas estejam valores longitudinais de 18.819 e 15.809 graus, respectivamente. A extensão espacial das planícies aluviais é um espaço que se estende por um plano de oeste a leste entre as duas latitudes e um plano de norte a sul entre as duas longitudes. Você pode incluir estes valores em um sistema de referência espacial, atribuindo-os a determinados parâmetros. Se a coluna espacial incluir coordenadas e medidas Z, será necessário incluir também as coordenadas e medidas Z maiores e menores no sistema de referência espacial.

O termo extensão espacial pode se referir não somente a uma expansão real de localizações, como no parágrafo anterior; mas também a uma possível expansão. Suponha que as planícies aluviais, no exemplo anterior, tivessem que ampliar pelos próximos cinco anos. Seria possível estimar quais seriam as coordenadas mais a oeste, mais a leste, mais ao norte e mais ao sul das planícies no final do quinto ano. Você, então, poderia atribuir essas estimativas, em vez das coordenadas atuais, aos parâmetros de uma extensão espacial. Dessa forma, você poderia manter o sistema de referência espacial à medida que as planícies se expandem e suas latitudes e longitudes maiores são incluídas na coluna espacial. Caso contrário, se o sistema de referência espacial estiver limitado às latitudes e longitudes originais, ele precisaria ser alterado ou substituído à medida que as planícies aluviais se expandirem.

Convertendo em valores que melhoram o desempenho

Geralmente, a maioria das coordenadas em um sistema de coordenadas são valores decimais; algumas são inteiros. Além disso, as coordenadas a leste da origem são positivas; as que estão a oeste são negativas. Antes de serem armazenadas pelo Spatial Extender, as coordenadas negativas são convertidas em valores positivos e as coordenadas decimais são convertidas em inteiros. Como resultado, todas as coordenadas são armazenadas pelo Spatial Extender como inteiros positivos. A finalidade é melhorar o desempenho quando as coordenadas são processadas.

Alguns parâmetros em um sistema de referência espacial são utilizados para fazer as conversões descritas no parágrafo precedente. Um parâmetro, chamado de *deslocamento*, é subtraído de cada coordenada negativa, que deixa um valor positivo como restante. Cada coordenada decimal é multiplicada por outro parâmetro, chamado de *fator de escala*, que resulta em um inteiro cuja precisão é igual à da coordenada decimal. (O deslocamento é subtraído de coordenadas positivas e de negativas; e as coordenadas não decimais, bem como as decimais, são multiplicadas pelo fator de escala. Dessa forma, todas as coordenadas positivas e não decimais permanecem correspondentes às decimais).

Estas conversões ocorrem internamente e permanecem em vigor até que as coordenadas sejam recuperadas. Os resultados de entrada e de consulta sempre contêm coordenadas em seu formato original, não convertido.

Decidindo o Uso de um Sistema de Referência Existente ou a Criação de um Novo Sistema

Responder à seguinte série de questões pode ajudá-lo na decisão do uso de um sistema de referência espacial existente ou da criação de um novo sistema.

Sobre Esta Tarefa

Depois de determinar qual sistema de coordenadas será utilizado, você estará pronto para fornecer um sistema de referência espacial que seja adequado aos dados de coordenadas com os quais você está trabalhando. O DB2 Spatial Extender fornece cinco sistemas de referência espacial para dados espaciais.

Procedimento

Para decidir entre o uso de um sistema de referência existente ou a criação de um novo sistema:

1. Responda às seguintes perguntas para determinar se é possível usar um dos sistemas de referência espacial existentes.
   a. O sistema de coordenadas no qual o sistema de referência espacial existente é baseado inclui a área geográfica com a qual você está trabalhando? Estes sistemas de coordenadas são mostrados em "Sistemas de Referência Espacial Fornecidos com o DB2 Spatial Extender" na página 43.
   b. Os fatores de conversão associados a um dos sistemas de referência espacial existentes trabalham com seus dados de coordenadas?

      O Spatial Extender utiliza valores de deslocamento e fatores de escala para converter os dados de coordenadas que você fornece para inteiros positivos. Para determinar se seus dados de coordenadas trabalham com os valores de compensação e os fatores de escala determinados para um dos sistemas de referência espacial existentes:

1) Reveja as informações em "Fatores de Conversão que Transformam Dados de Coordenadas em Inteiros" na página 46.
2) Veja como esses fatores são definidos para os sistemas de referência espacial existentes. Se, depois de aplicar o valor de deslocamento para as coordenadas mínimas X e Y, elas não forem ambas maiores que 0, você deverá criar um novo sistema de referência espacial e definir os deslocamentos manualmente. Para obter informações adicionais sobre como criar um novo sistema de referência espacial, consulte "Criando um Sistema de Referência Espacial" na página 47.

c. Os dados com os quais você está trabalhando incluem coordenadas de altura e profundidade (coordenadas Z) ou medidas (coordenadas M)? Se estiver trabalhando com coordenadas Z ou M, talvez seja necessário criar um novo sistema de referência espacial com deslocamentos Z ou M e fatores de escala adequados para seus dados.

2. Se os sistemas de referência espacial existentes não funcionarem com seus dados, será necessário "Criando um Sistema de Referência Espacial" na página 47.

O que Fazer Depois

Depois de decidir qual sistema de referência espacial é necessário, especifique esta opção para o Spatial Extender nas funções ou chamadas de procedimento.

Sistemas de Referência Espacial Fornecidos com o DB2 Spatial Extender

O sistema de referência espacial converte os dados de coordenadas em inteiros positivos.

O DB2 Spatial Extender fornece os sistemas de referência espacial que são mostrados na tabela a seguir, juntamente com o sistema de coordenadas no qual cada sistema de referência espacial é baseado e os valores de compensação e fatores de escala usados pelo DB2 Spatial Extender para converter os dados de coordenadas em números inteiros positivos. Você pode localizar informações sobre esses sistemas de referência na visualização de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Se você estiver trabalhando com graus decimais, os valores de deslocamento e os fatores de escala dos sistemas de referência espacial padrão suportarão o intervalo completo de coordenadas de latitude/longitude e manterão 6 posições decimais, equivalentes a aproximadamente 10 cm. Os dados do Mapa Espacial de Amostra e os dados de Referência de Geocodificador estão em graus decimais.

Se você planeja utilizar o geocodificador, que funciona apenas com endereços americanos, selecione ou crie um sistema de referência espacial que trate de coordenadas americanas, como o sistema de coordenadas GCS_NORTH_AMERICAN_1983. Se você não especificar de qual sistema de coordenadas seus dados espaciais devem ser derivados, o Spatial Extender utilizará o sistema de referência espacial DEFAULT_SRS.

Se nenhum sistema de referência espacial padrão atender suas necessidades, você poderá criar um novo.

Tabela 1. Sistemas de Referência Espacial Fornecidos com o DB2 Spatial Extender

Sistema de referência espacial	ID SRS	Sistema de coordenadas	Valores de deslocamento	Fatores de escala	Quando utilizar
DEFAULT _SRS	0	Nenhum	xOffset = 0 yOffset = 0 zOffset = 0 mOffset = 0	xScale = 1 yScale = 1 zScale = 1 mScale = 1	Você pode selecionar este sistema quando os dados são independentes de um sistema de coordenadas ou você não pode ou não precisa especificar um.
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	Você poderá selecionar esse sistema de referência espacial se planeja utilizar os dados de amostra americanos fornecidos com o DB2 Spatial Extender. Se os dados de coordenadas com os quais está trabalhando foram coletados após 1983, utilize este sistema em lugar do NAD27_SRS_1002.

Tabela 1. Sistemas de Referência Espacial Fornecidos com o DB2 Spatial Extender (continuação)

Sistema de referência espacial	ID SRS	Sistema de coordenadas	Valores de deslocamento	Fatores de escala	Quando utilizar
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	Você poderá selecionar esse sistema de referência espacial se planeja utilizar os dados de amostra americanos fornecidos com o DB2 Spatial Extender. Se os dados de coordenadas com os quais está trabalhando foram coletados após 1983, utilize este sistema em lugar de NAD83_SRS_1. Esse sistema fornece um grau de precisão maior que os outros sistemas de referência espacial padrão.
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	Você poderá selecionar esse sistema de referência espacial se estiver trabalhando com dados fora dos Estados Unidos (Este sistema manipula coordenadas no mundo todo). Não o utilize se planeja usar o geocodificador padrão fornecido com o DB2 Spatial Extender, pois ele se destina somente a endereços americanos.

Tabela 1. Sistemas de Referência Espacial Fornecidos com o DB2 Spatial Extender (continuação)

Sistema de referência espacial	ID SRS	Sistema de coordenadas	Valores de deslocamento	Fatores de escala	Quando utilizar
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	Esse sistema de referência espacial baseia-se em um sistema de coordenadas para endereços alemães.

Fatores de Conversão que Transformam Dados de Coordenadas em Inteiros

O DB2 Spatial Extender utiliza valores de deslocamento e fatores de escala para converter os dados de coordenadas que você fornece em inteiros positivos. Os sistemas de referência espacial padrão já possuem valor de deslocamento e fatores de escala associados a eles. Se você estiver criando um novo sistema de referência espacial, determine os fatores de escala e, opcionalmente, os valores de deslocamento que funcionam melhor com seus dados.

Valores de deslocamento

Um valor de deslocamento é um número que é subtraído de todas as coordenadas, deixando somente valores positivos como restante.

O Spatial Extender converte seus dados de coordenadas utilizando as seguintes fórmulas para assegurar que todos os valores de coordenadas ajustados sejam maiores que 0.

Notação da fórmula: Nestas fórmulas, a notação "min" representa "o mínimo de todos". Por exemplo, "min(x)" significa "o mínimo de todas as coordenadas x". O deslocamento para cada direção geográfica é representado como dimensionOffset. Por exemplo, xOffset é o valor de deslocamento aplicado a todas as coordenadas X.

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

Fatores de escala

Um fator de escala é um valor que, quando multiplicado por coordenadas e medidas decimais, resulta em inteiros com pelo menos o mesmo número de dígitos significativos das coordenadas e medidas originais.

O Spatial Extender converte seus dados de coordenadas decimais utilizando as seguintes fórmulas para assegurar que todos os valores de coordenadas ajustados sejam inteiros positivos. Os valores convertidos não podem exceder 2^{53} (aproximadamente $9 * 10^{15}$).

Notação da fórmula: Nestas fórmulas, a notação "max" representa "o máximo de todos". O deslocamento de cada dimensão geográfica é representado como um deslocamento de dimensão (por exemplo, xOffset é o valor de deslocamento

aplicado a todas as coordenadas X). O fator de escala de cada dimensão geográfica é representado como Escala de dimensão (por exemplo, xScale é o fator de escala aplicado a coordenadas X).

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

Ao escolher quais fatores de escala funcionam melhor com os dados de coordenadas, certifique-se de:

- Utilizar o mesmo fator de escala para as coordenadas X e Y.
- Quando multiplicado por uma coordenada X decimal ou uma coordenada Y decimal, o fator de escala gera um valor menor que 2^{53}. Uma técnica comum é tornar o fator de escala uma potência de 10. Ou seja, o fator de escala deve ser 10 para a primeira potência (10), 10 para a segunda potência (100), 10 para a terceira potência (1000), ou, se necessário, um fator maior.
- O fator de escala é grande o suficiente para assegurar que o número de dígitos significativos no novo inteiro seja igual ao número de dígitos significativos na coordenada decimal original.

Exemplo

Suponha que a função ST_Point receba uma entrada que consiste em uma coordenada X de 10.01, em uma coordenada Y de 20.03 e no identificador de um sistema de referência espacial. Quando ST_Point é chamado, ele multiplica o valor de 10,01 e o valor de 20,03 pelo fator de escala do sistema de referência espacial para as coordenadas X e Y. Se esse fator de escala for 10, os inteiros resultantes que o Spatial Extender armazenará serão 100 e 200, respectivamente. Como o número de dígitos significativos nestes inteiros (3) é menor que o número de dígitos significativos nas coordenadas (4), o Spatial Extender não poderá converter estes inteiros novamente para as coordenadas originais ou derivar deles valores que sejam consistentes com o sistema de coordenadas ao qual estas coordenadas pertencem. Porém, se o fator de escala for 100, os inteiros resultantes que o DB2 Spatial Extender armazena serão 1001 e 2003, valores que podem ser convertidos de novo nas coordenadas originais ou dos quais as coordenadas compatíveis podem ser derivadas.

Unidades para Valores de Deslocamento e Fatores de Escala

Caso você utilize um sistema de referência espacial existente ou crie um novo, as unidades dos valores de deslocamento e os fatores de escala variarão dependendo do tipo de sistema de coordenadas que estiver utilizando.

Por exemplo, se estiver utilizando um sistema de coordenadas geográficas, os valores estarão em unidades angulares, como graus decimais; se estiver utilizando um sistema de coordenadas projetadas, os valores estarão em unidades lineares, como metros ou pés.

Criando um Sistema de Referência Espacial

Crie um novo sistema de referência espacial se nenhum dos sistemas de referência espacial fornecidos com o DB2 Spatial Extender funcionar com seus dados.

Procedimento

Para criar um sistema de referência espacial:

1. Escolha um ID do sistema de referência espacial (SRID) que ainda não esteja em uso.
2. Decida sobre o grau de precisão para o sistema de referência espacial, usando um dos seguintes métodos:
   - Indicando as extensões da área geográfica com a qual você está trabalhando e os fatores de escala que deseja usar com seus dados de coordenadas. O Spatial Extender calcula os valores de compensação.
   - Indicando os valores de compensação (necessários para o Spatial Extender converter valores negativos em valores positivos) e fatores de escala (necessários para o Spatial Extender converter valores decimais em números inteiros). Use este método quando precisar seguir critérios estritos para exatidão ou precisão.
3. Calcule as informações de conversão requeridas pelo Spatial Extender para converter dados de coordenadas em números inteiros positivos. As informações para cálculo variam, dependendo do método escolhido em 2.
   - Para o método de extensões, calcule as seguintes informações:
     – *Fatores de escala*. Se qualquer uma das coordenadas com a qual você está trabalhando for valor decimal, calcule fatores de escala. Os fatores de escala são números que, quando multiplicados por coordenadas e medidas decimais, resultam em inteiros com pelo menos o mesmo número de dígitos significativos das coordenadas e medidas originais. Se as coordenadas forem números inteiros, os fatores de escala poderão ser definidos em 1. Se as coordenadas forem valores decimais, configure o fator de escala como um número que converte a parte decimal em um valor de número inteiro. Por exemplo, se as unidades de coordenadas forem metros e a precisão dos dados for de 1 cm, você precisará de um fator de escala de 100.
     – *Valores mínimos e máximos* para suas coordenadas e medidas.
   - Para o método de compensação, calcule as seguintes informações:
     – *Valores de compensação*. Se os dados de coordenadas incluírem números ou medidas negativas, especifique os valores de compensação que deseja usar. Um deslocamento é um número que é subtraído de todas as coordenadas, deixando somente valores positivos como restante. Se estiver trabalhando com coordenadas positivas, defina todos os valores de deslocamento como 0. Se não estiver trabalhando com coordenadas positivas, selecione um deslocamento que, quando aplicado nos dados de coordenadas, resulte em inteiros menores que o maior valor inteiro positivo (9,007,199,254,740,992).
     – *Fatores de escala*. Se alguma coordenada das localizações que estiver representando for um número decimal, determine quais fatores de escala serão utilizados e digite esses fatores de escala na janela Criar Sistema de Referência Espacial.
4. Emita o comando **db2se create_srs** ou chame o procedimento DB2SE.ST_CREATE_SRS para criar o sistema de referência espacial.

   O exemplo a seguir mostra como criar um sistema de referência espacial chamado mysrs:

```
db2se create_srs mydb -srsName mysrs -srsID 100
         -xScale 10 -coordsysName GCS_North_American_1983
```

Calculando Fatores de Escala

Se você criar um sistema de referência espacial e qualquer uma das coordenadas com as quais você está trabalhando for valor decimal, calcule os fatores de escala

apropriados para suas coordenadas e medidas. Os fatores de escala são números que, quando multiplicados por coordenadas e medidas decimais, resultam em inteiros com pelo menos o mesmo número de dígitos significativos das coordenadas e medidas originais.

Sobre Esta Tarefa

Após calcular fatores de escala, você precisa determinar os valores de extensão. Em seguida, submeta o comando **db2se create_srs** ou chame o procedimento DB2SE.ST_CREATE_SRS.

Procedimento

Para calcular os fatores de escala:

1. Determine quais coordenadas X e Y são, ou provavelmente serão, números decimais. Suponha, por exemplo, que das várias coordenadas X e Y com as quais você estará lidando, você determina que três delas são números decimais: 1.23, 5.1235 e 6.789.
2. Encontre a coordenada decimal que tenha a precisão decimal mais longa. Em seguida, determine por qual potência de 10 esta coordenada pode ser multiplicada para gerar um inteiro de igual precisão. Por exemplo, das três coordenadas decimais no exemplo atual, 5,1235 tem a precisão decimal mais longa. Multiplicá-lo por 10 à quarta potência (10000) gerará o inteiro 51235.
3. Determine se o inteiro produzido pela multiplicação que acaba de ser descrita é menor do que 2^{53}. 51235 não é muito grande. Mas, suponha que, além de 1.23, 5.11235 e 6.789, o intervalo de coordenadas X e Y inclua um quarto valor decimal, 10000000006.789876. Como esta precisão decimal da coordenada é maior do que as outras três, você pode multiplicar esta coordenada — não 5,1235 — por uma potência de 10. Para convertê-la em um inteiro, você poderia multiplicá-la por 10 elevado a seis (1000000). Mas o valor resultante, 10000000006789876, é maior do que 2^{53}. Se o DB2 Spatial Extender tentasse armazená-lo, os resultados seriam imprevisíveis.

   Para evitar este problema, selecione uma potência de 10 que, quando multiplicada pela coordenada original, gere um número decimal que o DB2 Spatial Extender possa truncar para um inteiro armazenável, com perda mínima de precisão. Neste caso, você pode selecionar 10 à quinta potência (100000). Multiplicando 100000 por 10000000006.789876 resulta em 1000000000678987.6. O DB2 Spatial Extender arredonda este número para 1000000000678988, reduzindo ligeiramente sua precisão.

Fatores de Conversão que Transformam Dados de Coordenadas em Inteiros

O DB2 Spatial Extender utiliza valores de deslocamento e fatores de escala para converter os dados de coordenadas que você fornece em inteiros positivos. Os sistemas de referência espacial padrão já possuem valor de deslocamento e fatores de escala associados a eles. Se você estiver criando um novo sistema de referência espacial, determine os fatores de escala e, opcionalmente, os valores de deslocamento que funcionam melhor com seus dados.

Determinando Coordenadas e Medidas Mínimas e Máximas

Determine as coordenadas e medidas mínimas e máximas se você especificar transformações de extensão ao criar um sistema de referência espacial.

Sobre Esta Tarefa

Depois de determinar os valores de extensões, se qualquer uma das coordenadas for valor decimal, será necessário calcular fatores de escala. Caso contrário, submeta o comando **db2se create_srs** ou chame o procedimento DB2SE.ST_CREATE_SRS.

Procedimento

Para determinar as coordenadas e medidas mínimas e máximas das localizações que deseja representar:

1. Determine as coordenadas X mínimas e máximas. Para encontrar a coordenada X mínima, identifique-a no seu domínio que esteja na extremidade oeste. (Se a localização ficar à oeste do ponto de origem, essa coordenada será um valor negativo). Para encontrar a coordenada X máxima, identifique-a no seu domínio que esteja na extremidade leste. Por exemplo, se estiver representando poços de petróleo e cada um estiver definido por um par de coordenadas X e Y, a coordenada X que indica a localização do poço que está na extremidade oeste será a coordenada X mínima e a que indica a localização na extremidade leste será a coordenada X mínima.

   Dica: Para tipos de recursos múltiplos, como polígonos múltiplos, verifique se escolheu o ponto mais extremo no polígono mais extremo na direção que está calculando. Por exemplo, se estiver tentando identificar a coordenada X mínima, identifique a coordenada na extremidade X oeste do polígono que está na extremidade oeste no polígono múltiplo.
2. Determine as coordenadas Y mínimas e máximas. Para encontrar a coordenada Y mínima, identifique-a no domínio que esteja na extremidade sul. (Se a localização ficar ao sul do ponto de origem, essa coordenada será um valor negativo). Para determinar a coordenada Y máxima, localize-a no domínio que esteja na extremidade norte.
3. Determine as coordenadas Z mínimas e máximas. A coordenada Z mínima é a maior das coordenadas de profundidade e a coordenada Z a maior das coordenadas de altura.
4. Determine as medidas mínimas e máximas. Se você for incluir medidas nos dados espaciais, determine qual medida tem o valor numérico mais alto e qual tem o mais baixo.

Calculando Valores de Deslocamento

Especifique valores de compensação se os dados de coordenadas incluírem números ou medidas negativas. Um deslocamento é um número que é subtraído de todas as coordenadas, deixando somente valores positivos como restante.

Sobre Esta Tarefa

Se criar um sistema de referência espacial e seus dados de coordenadas incluírem números ou medidas negativas, você deverá especificar os valores de compensação que deseja usar. Você pode aprimorar o desempenho de operações espaciais quando as coordenadas forem inteiros positivos em vez de números ou medidas negativas.

Procedimento

Para calcular valores de deslocamento para as coordenadas com que está trabalhando:

1. Determine as menores coordenadas X, Y e Z negativas dentro do intervalo de coordenadas das localizações que deseja representar. Se os seus dados tiverem que incluir medidas negativas, determine a menor destas medidas.
2. Opcional, mas recomendado: Indique ao DB2 Spatial Extender que o domínio que abrange os locais de seu interesse é maior do que realmente é. Assim, depois de gravar os dados sobre essas localizações em uma coluna espacial, você poderá incluir dados sobre localizações de recursos novos, à medida que forem incluídos em distâncias distantes do domínio, sem a necessidade de substituir o sistema de referência espacial por outro.

   Para cada coordenada e medida identificada na etapa 1, inclua uma quantidade com valor de cinco a dez por cento da coordenada ou medida. O resultado é referido como um *valor aumentado*. Por exemplo, se a menor coordenada X negativa for –100, você pode incluir –5 a ela, gerando um valor aumentado de –105. Posteriormente, quando você criar o sistema de referência espacial, indique que a menor coordenada X é -105, em vez do valor verdadeiro de –100. O DB2 Spatial Extender, então, interpretará –105 como o limite mais a oeste de seu domínio.
3. Encontre um valor que, quando subtraído do valor X aumentado, reste zero, este é o valor de deslocamento para as coordenadas X. O DB2 Spatial Extender subtrai esse número de todas as coordenadas X para produzir somente valores positivos.

   Por exemplo, se o valor X aumentado for –105, será necessário subtrair –105 dele para obter 0. O DB2 Spatial Extender irá, então, subtrair –105 de todas as coordenadas X que estão associadas aos recursos que estão sendo representados. Como nenhuma destas coordenadas é maior que –100, todos os valores que resultam da subtração serão positivos.
4. Repita a etapa 3 para o valor Y aumentado, o valor Z aumentado e a medida aumentada.

Criando um Sistema de Referência Espacial

Crie um novo sistema de referência espacial se nenhum dos sistemas de referência espacial fornecidos com o DB2 Spatial Extender funcionar com seus dados.

Procedimento

Para criar um sistema de referência espacial:

1. Escolha um ID do sistema de referência espacial (SRID) que ainda não esteja em uso.
2. Decida sobre o grau de precisão para o sistema de referência espacial, usando um dos seguintes métodos:
   - Indicando as extensões da área geográfica com a qual você está trabalhando e os fatores de escala que deseja usar com seus dados de coordenadas. O Spatial Extender calcula os valores de compensação.
   - Indicando os valores de compensação (necessários para o Spatial Extender converter valores negativos em valores positivos) e fatores de escala (necessários para o Spatial Extender converter valores decimais em números inteiros). Use este método quando precisar seguir critérios estritos para exatidão ou precisão.

3. Calcule as informações de conversão requeridas pelo Spatial Extender para converter dados de coordenadas em números inteiros positivos. As informações para cálculo variam, dependendo do método escolhido em 2 na página 48.
   - Para o método de extensões, calcule as seguintes informações:
     - *Fatores de escala*. Se qualquer uma das coordenadas com a qual você está trabalhando for valor decimal, calcule fatores de escala. Os fatores de escala são números que, quando multiplicados por coordenadas e medidas decimais, resultam em inteiros com pelo menos o mesmo número de dígitos significativos das coordenadas e medidas originais. Se as coordenadas forem números inteiros, os fatores de escala poderão ser definidos em 1. Se as coordenadas forem valores decimais, configure o fator de escala como um número que converte a parte decimal em um valor de número inteiro. Por exemplo, se as unidades de coordenadas forem metros e a precisão dos dados for de 1 cm, você precisará de um fator de escala de 100.
     - *Valores mínimos e máximos* para suas coordenadas e medidas.
   - Para o método de compensação, calcule as seguintes informações:
     - *Valores de compensação*. Se os dados de coordenadas incluírem números ou medidas negativas, especifique os valores de compensação que deseja usar. Um deslocamento é um número que é subtraído de todas as coordenadas, deixando somente valores positivos como restante. Se estiver trabalhando com coordenadas positivas, defina todos os valores de deslocamento como 0. Se não estiver trabalhando com coordenadas positivas, selecione um deslocamento que, quando aplicado nos dados de coordenadas, resulte em inteiros menores que o maior valor inteiro positivo (9,007,199,254,740,992).
     - *Fatores de escala*. Se alguma coordenada das localizações que estiver representando for um número decimal, determine quais fatores de escala serão utilizados e digite esses fatores de escala na janela Criar Sistema de Referência Espacial.
4. Emita o comando **db2se create_srs** ou chame o procedimento DB2SE.ST_CREATE_SRS para criar o sistema de referência espacial.

   O exemplo a seguir mostra como criar um sistema de referência espacial chamado mysrs:

```
db2se create_srs mydb -srsName mysrs -srsID 100
         -xScale 10 -coordsysName GCS_North_American_1983
```

Capítulo 8. Configurando Colunas Espaciais

Em preparação para obter dados espaciais para um projeto, você deve escolher um sistema de coordenadas e um sistema de referência espacial e, em seguida, fornecer uma ou mais colunas da tabela para conter os dados.

Visualização de Colunas Espaciais

Quando você usa uma ferramenta de visualização, como ArcExplorer para DB2, para consultar uma coluna espacial, a ferramenta retorna resultados na forma de exibição gráfica; por exemplo, um mapa de limites de áreas ou o layout de um sistema rodoviário.

Algumas ferramentas de visualização requerem que todas as linhas da coluna utilizem o mesmo sistema de referência espacial. A maneira que você aplica esta restrição é registrando a coluna com um sistema de referência espacial.

Tipos de dados espaciais

Quando você ativa um banco de dados para operações espaciais, o DB2 Spatial Extender fornece ao banco de dados uma hierarquia de tipos de dados estruturados.

A Figura 12 apresenta esta hierarquia. Nesta figura, os tipos que podem ser instanciados tem o plano de fundo em branco; os tipos que não podem ser instanciados possuem um plano de fundo sombreado.

Os tipos de dados instanciáveis são: ST_Point, ST_LineString, ST_Polygon, ST_GeomCollection, ST_MultiPoint, ST_MultiPolygon e ST_MultiLineString.

Os tipos de dados que não são instanciáveis são ST_Geometry, ST_Curve, ST_Surface, ST_MultiSurface e ST_MultiCurve.

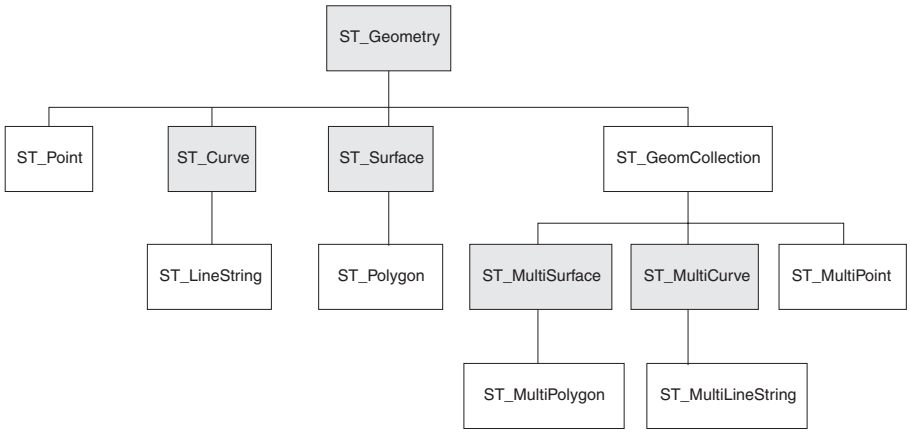


Figura 12. Hierarquia dos tipos de dados espaciais. Tipos de dados em caixas brancas podem ser instanciados. Os tipos de dados nomeados nas caixas sombreadas não são instanciáveis.

A hierarquia na Figura 12 inclui:

- Tipos de dados para recursos geográficos que podem ser captados formando uma única unidade; por exemplo, residências individuais e lagos isolados.
- Tipos de dados para recursos geográficos que são compostos de várias unidades ou componentes; por exemplo, sistemas de canais e grupos de ilhas em um lago.
- Um tipo de dados para recursos geográficos de todos os tipos.

Tipos de Dados para Recursos de uma Única Unidade

Utilize ST_Point, ST_LineString e ST_Polygon para armazenar coordenadas que definem o espaço ocupado por recursos que podem ser percebidos na formação de uma única unidade.

- Utilize ST_Point quando desejar indicar o ponto no espaço que é ocupado por um recurso geográfico separado. O recurso pode ser muito pequeno, como um poço de água; pode ser muito grande, como uma cidade; ou pode ser intermediário, como um conjunto de prédios ou um parque. Em cada caso, o ponto no espaço pode estar localizado na interseção de uma linha de coordenada leste-oeste (por exemplo, uma paralela) e uma norte-sul (por exemplo, um meridiano). Um item de dados ST_Point inclui uma coordenada X e uma coordenada Y que definem essa interseção. A coordenada X indica onde a interseção está situada na linha leste-oeste; a coordenada Y indica onde a interseção está situada na linha norte-sul.
- Utilize ST_Linestring para coordenadas que definem o espaço que é ocupado por recursos lineares; por exemplo, ruas, canais e canalizações.
- Utilize ST_Polygon quando desejar indicar a extensão do espaço coberto por um recurso multifacetado; por exemplo, uma região, uma floresta ou um habitat natural. Um item de dados de ST_Polygon consiste de coordenadas que definem o limite de tal recurso.

Em alguns casos, ST_Polygon e ST_Point podem ser utilizados para o mesmo recurso. Por exemplo, suponha que você precise de informações espaciais sobre um complexo de apartamentos. Se desejar representar o ponto no espaço onde cada prédio no complexo está localizado, utilize ST_Point para armazenar as coordenadas X e Y que definem cada um desses pontos. De outra forma, se deseja representar a área ocupada pelo complexo como um todo, utilize ST_Polygon para armazenar as coordenadas que definem o limite dessa área.

Tipos de Dados para Recursos de Várias Unidades

Utilize ST_MultiPoint, ST_MultiLineString e ST_MultiPolygon para armazenar coordenadas que definem espaços ocupados por recursos compostos de várias unidades.

- Utilize ST_MultiPoint quando estiver representando recursos compostos de unidades cujas localizações sejam referidas individualmente por uma coordenada X e uma coordenada Y. Por exemplo, considere uma tabela cujas linhas representam uma cadeia de ilhas. Foi identificada a coordenada X e a coordenada Y para cada ilha. Se deseja que a tabela inclua estas coordenadas e as coordenadas de cada cadeia como um todo, defina uma coluna ST_MultiPoint para conter estas coordenadas.
- Utilize ST_MultiLineString quando estiver representando recursos compostos de unidades lineares e desejar armazenar as coordenadas para as localizações destas unidades e a localização de cada recurso como um todo. Por exemplo, considere uma tabela cujas linhas representam um sistema de rios. Se deseja que a tabela inclua coordenadas para as localizações do sistema e de seus componentes, defina uma coluna ST_MultiLineString para conter estas coordenadas.

- Utilize ST_MultiPolygon quando estiver representando recursos compostos de unidades multifacetadas e desejar armazenar as coordenadas para as localizações destas unidades e a localização de cada recurso como um todo. Por exemplo, considere uma tabela cujas linhas representam áreas rurais e as fazendas em cada área. Se deseja que a tabela inclua coordenadas para as localizações das áreas e fazendas, defina uma coluna ST_MultiPolygon para conter estas coordenadas.

Multiunidade não significa uma coleção de entidades individuais. Multiunidade se refere a uma agregação das partes que compõem o todo.

Um Tipo de Dados para Todos os Recursos

Você pode utilizar ST_Geometry quando não tiver certeza sobre qual dos outros tipos de dados utilizar.

Como ST_Geometry é a raiz da hierarquia à qual os outros tipos de dados pertencem, uma coluna ST_Geometry pode conter o mesmo tipo de itens de dados que as colunas dos outros tipos de dados podem conter.

Atenção: Algumas ferramentas de visualização não suportam colunas ST_Geometry, mas apenas colunas às quais um subtipo adequado de ST_Geometry foi designado.

Criando Colunas espaciais

Você deve criar colunas espaciais para armazenar e recuperar dados espaciais.

Antes de Iniciar

Antes de criar uma coluna espacial, seu ID do usuário deve conter as autorizações necessárias para a instrução DB2 SQL CREATE TABLE ou ALTER TABLE. O ID do usuário deve ter pelo menos uma das seguintes autoridades ou privilégios:

- Autoridade DBADM no banco de dados no qual reside a tabela que contém a coluna
- Autoridade CREATETAB no banco de dados e o privilégio USE no espaço de tabelas, além de um dos seguintes:
  - Autoridade IMPLICIT_SCHEMA no banco de dados, se o esquema do índice não existir
  - Privilégio CREATEIN no esquema, se o nome do esquema do índice se referir a um esquema existente
- Privilégio ALTER na tabela a ser alterada
- Privilégio CONTROL na tabela a ser alterada
- Privilégio ALTERIN no esquema da tabela a ser alterada

Sobre Esta Tarefa

Esta tarefa faz parte de uma tarefa maior: "Configurando Recursos Espaciais para um Projeto." Depois de escolher um sistema de coordenadas e determinar qual sistema de referência espacial utilizar para seus dados, crie uma coluna espacial em uma tabela existente ou importe dados espaciais para uma nova tabela.

Procedimento

Para criar colunas espaciais:

Você pode fornecer a seu banco de dados colunas espaciais de uma dentre várias maneiras:

- Use a instrução CREATE TABLE SQL para criar uma tabela e para incluir uma coluna espacial nessa tabela.
- Use a instrução ALTER TABLE SQL para incluir uma coluna espacial em uma tabela existente.
- Se estiver importando dados espaciais de um arquivo de formas, utilize o DB2 Spatial Extender para criar uma tabela e para fornecer a esta tabela uma coluna para conter os dados. Consulte "Importando dados de formato para uma tabela nova ou existente".

O que Fazer Depois

A próxima tarefa de configuração de recursos espaciais é “Registrando Colunas Espaciais”.

Registrando Colunas Espaciais

Registrar uma coluna espacial cria uma limitação na tabela, se possível, para assegurar que todas as geometrias utilizem o sistema de referência espacial especificado.

Antes de Iniciar

Se você estiver usando o processador de linha de comandos db2se ou um programa de aplicativo, seu ID de usuário deverá reter a autoridade SYSADM ou DBADM no banco de dados.

Sobre Esta Tarefa

Talvez você queira registrar uma coluna espacial nas seguintes situações:

- Acesso por ferramentas de visualização

  Se desejar que determinadas ferramentas de visualização — por exemplo, ArcExplorer para DB2 — gerem exibições gráficas dos dados em uma coluna espacial, deve garantir a integridade dos dados da coluna. Pode-se fazer isso impondo uma limitação que requer que todas as linhas da coluna utilizem o mesmo sistema de referência espacial. Para impor esta limitação, registre a coluna, especificando seu nome e o sistema de referência espacial que se aplica a ela.
- Acesso por índices espaciais

  Utilize o mesmo sistema de coordenadas para todos os dados em uma coluna espacial na qual você deseja criar um índice para assegurar que o índice espacial retorne os resultados corretos. Você registra uma coluna espacial para limitar todos os dados para utilizarem o mesmo sistema de referência espacial e, de forma correspondente, o mesmo sistema de coordenadas.

Procedimento

Registre uma coluna espacial usando um dos métodos a seguir:

- Emita o comando db2se register_spatial_column.
- Execute um aplicativo que chama o procedimento DB2GSE.ST_REGISTER_SPATIAL_COLUMN.

Consulte a coluna SRS_NAME na visualização DB2GSE.ST_GEOMETRY_COLUMNS para verificar o sistema de referência espacial escolhido para uma coluna particular depois de registrar a coluna.

Capítulo 9. Ocupando colunas espaciais

Depois de criar colunas espaciais e de registrar as que serão acessadas por estas ferramentas de visualização, é possível preencher as colunas com dados espaciais, importando-os, usando um geocodificador para derivá-los de dados de negócios ou usando funções espaciais para criá-los ou derivá-los de dados de negócios ou de outros dados espaciais.

Sobre como Importar e Exportar Dados espaciais

Você pode usar o DB2 Spatial para trocar dados espaciais entre o banco de dados e as origens de dados externas. Mais precisamente, você pode importar dados espaciais de origens externas transferindo-os para seu banco de dados em arquivos, chamados arquivos de troca de dados. Você também pode exportar dados espaciais de seu banco de dados para arquivos de troca de dados a partir dos quais as origens externas podem adquiri-los. Esta seção apresenta algumas das razões para importar e exportar dados espaciais e descreve a natureza dos arquivos de troca de dados suportados pelo DB2 Spatial Extender.

Razões para importar e exportar dados espaciais

Ao importar dados espaciais, você pode obter uma grande quantidade de informações espaciais que já estão disponíveis na indústria. Exportando-os você pode disponibilizá-los em um formato de arquivo padrão para aplicativos existentes. Considere estes cenários:

- Seu banco de dados contém dados espaciais que representam seus escritórios de vendas, clientes e outros assuntos comerciais. Você deseja suplementar estes dados com dados espaciais que representam seu ambiente cultural da organização — cidades, ruas, pontos de interesse, e assim por diante. Os dados que você deseja estão disponíveis a partir de um mapa do fornecedor. Você pode utilizar o DB2 Spatial Extender para importá-los de um arquivo de troca de dados oferecido pelo fornecedor.
- Você deseja migrar dados espaciais de um sistema Oracle para o ambiente do DB2. Você continuou utilizando um utilitário Oracle para gravar os dados em um arquivo de troca de dados. Utilize, então, o DB2 Spatial Extender para importar os dados deste arquivo para o banco de dados que foi ativado para operações espaciais.
- Você não está conectado ao DB2, e deseja utilizar um geonavegador para mostrar aos clientes apresentações visuais de informações espaciais. O navegador precisa apenas de arquivos para trabalhar; ele não precisa estar conectado a um banco de dados. É possível utilizar o DB2 Spatial Extender para exportar os dados para um arquivo de troca de dados e, então, utilizar um navegador para processar os dados em formato visual.

Arquivos de Formato

O DB2 Spatial Extender suporta arquivos de formato para troca de dados. O termo arquivo shape significa um conjunto de arquivos com o mesmo nome, mas com extensões de arquivo diferentes. A coleção pode incluir até quatro arquivos. Eles são:

- Um arquivo que contém dados espaciais em formato shape, um formato padrão da indústria desenvolvido pelo ESRI. Esses dados geralmente são chamados de formato de dados. A extensão de um arquivo que contém formato de dados é .shp.
- Um arquivo que contém dados de negócios que pertence a localizações definidas por formato de dados. A extensão deste arquivo é .dbf.
- Um arquivo que contém um índice para formato de dados. A extensão deste arquivo é .shx.
- Um arquivo que contém uma especificação do sistema de coordenadas no qual os dados em um arquivo .shp estão baseados. A extensão deste arquivo é .prj.

Os arquivos shape geralmente são utilizados para importar dados originados em sistemas de arquivos e para exportar dados para arquivos dentro de sistemas de arquivos.

Ao utilizar o DB2 Spatial Extender para importar dados de formato, você recebe pelo menos um arquivo .shp. Na maioria dos casos, você também pode receber um ou mais dos outros três tipos de arquivos shape.

Importando Dados de Formato para uma Tabela Nova ou Existente

Você pode importar dados de shape para uma tabela ou exibição existente ou pode criar uma tabela e importar dados de shape para ela em uma única operação.

Antes de Iniciar

Antes de importar dados da forma para uma tabela ou visualização existente, seu ID do usuário deve conter uma das seguintes autoridades ou privilégios:

- Autoridade DATAACCESS no banco de dados que contém a tabela ou visualização
- Privilégio CONTROL na tabela ou visualização
- Privilégio INSERT na tabela ou visualização
- Privilégio SELECT na tabela ou visualização (necessário apenas se a tabela incluir uma coluna de ID que não seja uma coluna IDENTITY)

Mais um dos seguintes privilégios:

- Privilégios para acessar os diretórios aos quais os arquivos de entrada e arquivos de erros pertencem
- Privilégios de leitura nos arquivos de entrada e privilégios de gravação em arquivos de erros

Antes de começar a criar uma tabela automaticamente e importar dados da forma para ela, seu ID do usuário deve conter as seguintes autoridades ou privilégios:

- Autoridade DBADM no banco de dados que conterá a tabela
- Autoridade CREATETAB no banco de dados que conterá a tabela
- Se o esquema já existir, o privilégio CREATEIN no esquema ao qual a tabela pertence
- Se o esquema especificado para a tabela não existir, a autoridade IMPLICIT_SCHEMA no banco de dados que contém a tabela

Mais um dos seguintes privilégios:

- Privilégios para acessar os diretórios aos quais os arquivos de entrada e arquivos de erros pertencem
- Privilégios de leitura nos arquivos de entrada e privilégios de gravação em arquivos de erros

Sobre Esta Tarefa

Escolha um dos seguintes métodos para importar dados sobre formas:

- Importar os dados de shape para uma coluna espacial em uma tabela existente, uma visualização atualizável existente ou uma visualização existente na qual um acionador INSTEAD OF para INSERTs está definido.
- Criar automaticamente uma tabela com uma coluna espacial e importar os dados de shape para esta coluna.

Procedimento

Para importar dados de forma para uma tabela nova ou existente:

Escolha o método que deseja utilizar para importar dados de shape:

- Emita o comando db2se import_shape.
- Execute um aplicativo que chama o procedimento DB2GSE.ST_IMPORT_SHAPE.

Exportando Dados para um Arquivo modelo

Você pode exportar dados espaciais retornados em resultados de consultas para um arquivo modelo.

Antes de Iniciar

Antes de exportar dados para um arquivo modelo, seu ID de usuário deve conter os seguintes privilégios:

- O privilégio para executar uma subseleção que retorna os resultados que você deseja exportar
- O privilégio para gravar no diretório em que reside o arquivo para o qual você exportará dados
- O privilégio para criar um arquivo para conter os dados exportados (obrigatório se o arquivo ainda não existir)

Para saber quais são estes privilégios e como obtê-los, consulte o administrador do banco de dados.

Sobre Esta Tarefa

Os dados espaciais que você exporta para um shapefile podem vir de várias origens, como uma tabela base, uma junção ou união de várias tabelas, conjuntos de resultados retornados quando você consulta suas visualizações ou a saída de uma função espacial.

Se existir um arquivo para o qual você deseja exportar dados, o DB2 Spatial Extender poderá anexar os dados a este arquivo. Se este arquivo não existir, você poderá utilizar o DB2 Spatial Extender para criar um.

Procedimento

Para exportar dados para um arquivo de forma:

Escolha um método para exportar dados para um shapefile:

- Emita o comando db2se export_shape a partir do processador de linha de comandos do db2se.
- Execute um aplicativo que chama o procedimento DB2GSE.ST_EXPORT_SHAPE.

Como Utilizar um Geocodificador

O uso de um geocodificador envolve a definição do trabalho que você deseja que seja executado por um geocodificador, a configuração do geocodificador e a execução do geocodificador no modo em lote.

Geocodificadores e Codificação Geográfica

Os termos geocodificador e codificação geográfica são utilizados em vários contextos. Esta discussão classifica estes contextos, para que os significados dos termos possam ser compreendidos sempre que você consultá-los. A discussão define geocodificador e codificação geográfica, descreve os modos nos quais um geocodificador opera, descreve uma maior atividade à qual a codificação geográfica pertence e resume as tarefas do usuário relacionadas à codificação geográfica.

No DB2 Spatial Extender, um geocodificador é uma função escalar que converte dados existentes (a entrada da função) em dados que é possível compreender em termos espaciais (a saída da função). Geralmente, os dados existentes são dados relacionais que descrevem ou nomeiam uma localização. O DB2 Spatial Extender pode suportar geocodificadores fornecidos pelo fornecedor e pelo usuário.

Um geocodificador fornecido pelo fornecedor pode converter endereços em coordenadas que o DB2 não armazena, mas grava em um arquivo. O outro pode converter o número de um escritório em um edifício comercial em coordenadas que definem a localização do escritório no edifício ou pode converter o identificador de uma prateleira em um armazém em coordenadas que definem a localização da prateleira no armazém.

Em outros casos, os dados existentes convertidos por um geocodificador podem ser dados espaciais. Por exemplo, um geocodificador fornecido pelo usuário pode converter coordenadas X e Y em dados que estão de acordo com um dos tipos de dados do DB2 Spatial Extender.

No DB2 Spatial Extender, geocodificação é apenas a operação na qual um geocodificador converte sua entrada em saída: converter endereços em coordenadas, por exemplo.

Modos

Um geocodificador opera em dois modos:

- No modo em lote, um geocodificador tenta, em uma única operação, converter todas as entradas de uma única tabela (ou, como alternativa, todos os endereços em um subconjunto de linhas especificado na tabela).

- No modo automático, um geocodificador converte dados assim que são inseridos ou atualizados em uma tabela. O geocodificador é ativado pelos disparos INSERT e UPDATE que estão definidos na tabela.

Processos de Codificação Geográfica

Geocodificação é uma das diversas operações pelas quais o conteúdo de uma coluna espacial em uma tabela do DB2 é derivado de outros dados. Esta discussão se refere a estas operações coletivamente como um processo de codificação geográfica . Os processos de codificação geográfica podem variar de geocodificador para geocodificador. Um geocodificador pode procurar arquivos de endereços conhecidos para determinar se cada endereço que recebe como entrada corresponde a um endereço conhecido em um determinado grau. Como os endereços conhecidos são semelhantes ao material de referência que as pessoas procuram quando fazem pesquisa, estes endereços são coletivamente chamados de dados de referência. Outros geocodificadores podem não precisar de dados de referência; eles podem verificar sua entrada de outras formas.

As tarefas do usuário

No DB2 Spatial Extender, as tarefas relacionadas à geocodificação são:

- Prescrever como determinadas partes do processo de codificação geográfica devem ser executadas para uma coluna espacial especificada; por exemplo, definir o grau mínimo ao qual os nomes de ruas em registros de entrada e nomes de ruas em dados de referência devem corresponder; definir o grau mínimo ao qual os endereços em registros de entrada e os endereços em dados de referência devem corresponder; e determinar quantos registros devem ser processados antes de cada consolidação. Esta tarefa pode ser referida como configurando a codificação geográfica ou configurando operações de codificação geográfica.
- Especificar que deve ser efetuado o geocode automático dos dados sempre que eles forem incluídos ou atualizados em uma tabela. Quando ocorre a codificação geográfica automática, as instruções que o usuário especificou ao configurar as operações de codificação geográfica serão efetivadas (exceto as instruções que envolvem consolidações; elas se aplicam somente à codificação geográfica em lote). Esta tarefa é referida como configurando em geocodificador para execução automática.
- Executando um geocodificador no modo em lote. Se o usuário já configurou operações de codificação geográfica, suas instruções permanecerão em efeito durante cada sessão em lote, a menos que o usuário as substitua. Se o usuário não configurou operações de codificação geográfica antes de uma determinada sessão, ele poderá especificar que elas devem ser efetivadas, configurando-as para essa sessão específica. Esta tarefa pode ser referida como executar um geocodificador no modo em lote e executar codificação geográfica no modo em lote.

Configurando Operações de Codificação Geográfica

O DB2 Spatial Extender permite definir, antecipadamente, o trabalho que precisa ser feito quando um geocodificador é chamado.

Antes de Iniciar

Antes de configurar as operações de geocodificação para um geocodificador específico, seu ID do usuário deve conter uma das seguintes autoridades ou privilégios:

- Autoridade DATAACCESS no banco de dados que contém as tabelas nas quais o geocodificador operará
- Privilégio CONTROL em cada tabela na qual o geocodificador opera
- Privilégios SELECT e UPDATE em cada tabela na qual o geocodificador opera

Sobre Esta Tarefa

É possível especificar os seguintes parâmetros quando um geocodificador é chamado:

- Para qual coluna o geocodificador deve fornecer dados.
- Se a entrada que o geocodificador lê a partir de uma tabela ou exibição deve ser limitada a um subconjunto de linhas na tabela ou exibição.
- O intervalo ou o número de registros que o geocodificador deve geocodificar em sessões em lote em uma unidade de trabalho.
- Requisitos para operações específicas de geocodificador. Por exemplo, um geocodificador pode efetuar geocode apenas nos registros que correspondem a suas contrapartes nos dados de referência em um grau especificado ou maior. Este grau é chamado de score mínimo de correspondência.

Você deve especificar os parâmetros na lista anterior antes de configurar o geocodificador para execução em modo automático. Desse ponto em diante, sempre que o geocodificador for chamado (não apenas automaticamente, mas também para execuções em lote), as operações de codificação geográfica serão executadas de acordo com suas especificações. Por exemplo, se você especificar que deve ser efetuado geocode em 45 registros no modo em lote em cada unidade de trabalho, será emitida uma consolidação depois de ser efetuado o geocode a cada quarenta e cinco registros. (Exceção: Você pode substituir suas especificações para sessões individuais de geocodificação em lote).

Você não precisa estabelecer padrões para operações de codificação geográfica antes de executar o geocodificador no modo em lote. Em vez disso, no momento em que iniciar uma sessão em lote, você pode especificar como as operações devem ser realizadas durante toda a execução. Se você estabelecer padrões para sessões em lote, poderá substituí-las, conforme necessário, por sessões individuais.

Procedimento

Para configurar operações de codificação geográfica:

Escolha de que forma você deseja configurar operações de geocodificação:

- Emita o comando db2se setup_gc.
- Execute um aplicativo que chama o procedimento DB2GSE.ST_SETUP_GEOCODING.

O que Fazer Depois

Recomendações: Quando um geocodificador lê um registro de dados do endereço, ele tenta fazer a correspondência desse registro com uma contraparte nos dados de referência. Explicando mais detalhadamente: a forma que ele procede é a seguinte: Primeiro, ele pesquisa nos dados de referência as ruas cujo CEP seja igual ao CEP do registro. Se ele encontra um nome de rua que seja semelhante ao do registro em um grau mínimo especificado, ou em um grau maior do que este mínimo, ele continua procurando um endereço inteiro. Se ele encontra um endereço inteiro que

seja semelhante ao do registro em um grau mínimo especificado, ou em um grau maior do que este mínimo, ele efetua geocode no registro. Se ele não encontra o endereço, retorna nulo.

O grau mínimo ao qual os nomes de ruas devem corresponder é referido como sensibilidade de ortografia. O grau mínimo ao qual todos os endereços devem corresponder é chamado de score mínimo de correspondência. Por exemplo, se a sensibilidade de ortografia for de 80, a correspondência entre os nomes de ruas deve conter, pelo menos, 80 por cento de precisão antes que o geocodificador pesquise o endereço inteiro. Se o score mínimo de correspondência for 60, a correspondência entre os endereços deve ser, pelo menos, 60 por cento de precisão antes que o geocodificador efetue geocode do registro.

Você pode especificar qual deve ser a sensibilidade de ortografia e o score mínimo de correspondência. Lembre-se de que você precisa ajustá-los. Por exemplo, suponha que a sensibilidade de ortografia e o score mínimo de correspondência sejam 95. Se os endereços nos quais você deseja efetuar geocode não foram validados com atenção, as correspondências com 95 por cento de precisão provavelmente não serão encontradas. Como resultado, o geocodificador provavelmente retornará nulo quando processar estes registros. Neste caso, é recomendável reduzir a sensibilidade de ortografia e o score mínimo de correspondência e executar o geocodificador novamente. Os scores recomendados para sensibilidade de ortografia e o score mínimo de correspondência são 70 e 60, respectivamente.

Conforme mencionado no início desta discussão, você pode determinar se a entrada que o geocodificador lê, a partir de uma tabela ou exibição, deve ser limitada a um subconjunto de linhas na tabela ou exibição. Por exemplo, considere os seguintes cenários:

- Você chama o geocodificador para efetuar geocode nos endereços em uma tabela no modo em lote. Infelizmente, o score mínimo de correspondência é muito alto, fazendo o geocodificador retornar nulo quando ele processa a maioria dos endereços. Você reduz o score mínimo de correspondência quando executar o geocodificador novamente. Para limitar sua entrada aos endereços nos quais não foi efetuado geocode, especifique se ele deve selecionar somente as linhas que contenham o nulo retornado anteriormente.
- O geocodificador seleciona somente as linhas que foram incluídas depois de uma determinada data.
- O geocodificador seleciona somente as linhas que contêm endereços em uma determinada área; por exemplo, um bloco de regiões ou um estado.

Conforme mencionado no início desta discussão, você pode determinar o número de registros que o geocodificador deve processar em sessões em lote em uma unidade de trabalho. Você pode fazer o geocodificador processar o mesmo número de registros em cada unidade de trabalho ou pode fazer ele processar todos os registros de uma tabela em uma única unidade de trabalho. Se você escolher a última alternativa, lembre-se de que:

- Você tem menos controle sobre o tamanho da unidade de trabalho do que os recursos alternativos anteriores. Consequentemente, você não pode controlar quantos bloqueios estão retidos ou quantas entradas de log são criadas durante a operação do geocodificador.
- Se o geocodificador encontrar um erro que precise de uma reversão, será necessário executar o geocodificador para executar todos os registros novamente.

O custo resultante com recursos pode ser caro se a tabela for extremamente grande e o erro e a reversão ocorrerem após a maioria dos registros ter sido processada.

Configurando um geocodificador para Execução Automática

Você pode configurar um geocodificador para converter dados automaticamente assim que os dados são incluídos ou atualizados em uma tabela.

Antes de Iniciar

Antes de configurar um geocodificador para execução automática:

- Você deve configurar as operações de codificação geográfica para cada coluna espacial que deve ser ocupada pela saída do geocodificador.
- Seu ID do usuário deve conter as seguintes autoridades ou privilégios:
  - Autoridades UDBADM e DATAACCESS no banco de dados que contêm a tabela na qual os acionadores que invocam o geocodificador serão definidos
  - Privilégio CONTROL
  - Privilégios ALTER, SELECT e UPDATE.
  - Os privilégios necessários para criar disparos nesta tabela.

Sobre Esta Tarefa

Você pode configurar um geocodificador para execução automática antes de chamá-lo no modo em lote. Portanto, é possível que a codificação geográfica automática preceda a codificação geográfica em lote. Se isso ocorrer, a codificação geográfica em lote provavelmente envolverá o processamento dos mesmos dados que foram processados automaticamente. Esta redundância não resultará em duplicação de dados porque, quando os dados espaciais são gerados duas vezes, a segunda geração de dados substitui a primeira. De qualquer modo, ele pode degradar o desempenho.

Antes de decidir se efetuará o geocode nos dados de endereços em uma tabela no modo em lote ou no modo automático, considere que:

- O desempenho é melhor na codificação geográfica em lote do que na codificação geográfica automática. Uma sessão em lote é aberta com uma inicialização e termina com uma limpeza. Na codificação geográfica automática, é efetuado geocode em cada item de dados em uma única operação que começa com inicialização e é concluída com uma limpeza.
- Em geral, uma coluna espacial ocupada por meio da codificação geográfica automática provavelmente seja mais atualizada do que uma coluna espacial ocupada por meio da codificação geográfica em lote. Depois de uma sessão em lote, os dados de endereço podem ficar acumulados e permanecer sem geocode até a próxima sessão. Mas, se o geocoding automático já estiver ativado, será efetuado o geocode nos dados de endereço assim que forem armazenados no banco de dados.

Procedimento

Para configurar um geocodificador para que seja executado automaticamente:

Escolha qual método utilizar para configurar geocodificação automática:

- Emita o comando db2se enable_autogc.

- Execute um aplicativo que chama o procedimento DB2GSE.ST_ENABLE_AUTOGEOCODING.

Executando um geocodificador no Modo em Lote

Quando um geocodificador é executado em modo em lote, múltiplos registros são convertidos em dados espaciais que entram em uma coluna específica.

Antes de Iniciar

Antes de executar um geocodificador no modo em lote, seu ID do usuário deve conter as seguintes autoridades ou privilégios:

- Autoridade DATAACCESS no banco de dados que contém a tabela cujos dados devem ser geocodificados
- Privilégio CONTROL nesta tabela
- Privilégio UPDATE nesta tabela

Você também precisa do privilégio SELECT nesta tabela para que possa especificar o número de registros a serem processados antes de cada consolidação. Se você especificar cláusulas WHERE para limitar as linhas nas quais o geocodificador deve operar, também poderá requerer o privilégio SELECT nas tabelas e exibições que forem referidas nestas cláusulas. Consulte o administrador do banco de dados.

Sobre Esta Tarefa

A qualquer momento, antes de executar um geocodificador para ocupar uma determinada coluna espacial, você pode configurar as operações de codificação geográfica para essa coluna. Configurar as operações envolve a especificação de alguns requisitos que devem ser atendidos quando o geocodificador for executado. Por exemplo, suponha que você exija que o DB2 Spatial Extender emita uma consolidação a cada 100 registros de entrada que forem processados pelo geocodificador. Ao configurar as operações, especifique 100 como o número requerido.

Quando estiver pronto para executar o geocodificador, você poderá substituir qualquer um dos valores especificados durante a configuração das operações. Suas substituições permanecerão em efeito somente durante a execução.

Se você não configurar as operações, sempre que estiver pronto para executar o geocodificador, deverá especificar como os requisitos devem ser atendidos durante a execução.

Procedimento

Para executar um geocodificador no modo em lote:

Escolha como chamar um geocodificador para ser executado em modo em lote:

- Emita o comando db2se run_gc.
- Execute um aplicativo que chama o procedimento DB2GSE.ST_RUN_GEOCODING.

Capítulo 10. DB2 Spatial Extender em um Ambiente de Banco de Dados Particionado

O DB2 Spatial Extender pode ser usado efetivamente em um ambiente de banco de dados particionado para executar uma análise espacial com relação a grandes tabelas de dados do cliente. Os ambientes de banco de dados particionados são úteis para executar aplicativos de data warehousing que podem ser criados com o IBM InfoSphere Warehouse.

Os ambientes de banco de dados particionado suportam o processamento de banco de dados de alto desempenho e altamente escalável por processamento paralelo através de um conjunto de nós. Cada nó tem seu próprio processador, memória e armazenamento em disco.

Em um ambiente de banco de dados particionado, você tem três alternativas para armazenar tabelas DB2 que contêm colunas espaciais. É possível colocá-las em um nó único, distribuí-las ao longo de múltiplos nós ou substituí-las ao longo de múltiplos nós. Índices espaciais são suportados para cada alternativa. A alternativa que você seleciona depende do tamanho das tabelas e de como elas serão utilizadas em consultas espaciais. Para obter informações adicionais sobre como particionar um banco de dados, consulte "Ambientes de Banco de Dados Particionado" em *Database Administration Concepts and Configuration Reference*.

Criando e Carregando Dados Espaciais em um Ambiente de Banco de Dados Particionado

A tabela particionada deve ter uma chave de particionamento definida que é utilizada para controlar a distribuição de linhas entre as partições.

Antes de Iniciar

Certifique-se de que o banco de dados esteja ativado para processamento espacial criando tabelas de catálogos espaciais e criando os tipos e funções espaciais no banco de dados. Consulte "Ativando um Banco de Dados para Operações espaciais" na página 30 para obter informações adicionais.

Sobre Esta Tarefa

Para desempenho ideal, certifique-se de que as tabelas que contêm colunas espaciais sejam distribuídas uniformemente entre partições. O algoritmo hashing padrão e o mapa de partição fazem isso.

Procedimento

Para obter melhores resultados, especifique uma ou mais colunas na instrução CREATE TABLE como a chave de particionamento. Se você não especificar uma chave de particionamento, o DB2 seleciona a primeira coluna numérica ou de caracteres como a chave de particionamento por padrão. Esse padrão pode não distribuir as linhas uniformemente entre partições.

Exemplo

O exemplo a seguir mostra como criar uma tabela para conter dados espaciais e usar o utilitário de importação do DB2 Spatial Extender para especificar a chave de particionamento:

```
CREATE TABLE myschema.counties (id INTEGER PRIMARY KEY,
   name VARCHAR(20),
   geom db2gse.st_polygon)
   IN nodestbs
   DISTRIBUTE BY HASH(id);

db2se import_shape mydb
-tableName counties
-tableSchema myschema
-spatialColumn shape
-fileName /shapefiles/counties.shp
-messagesFile counties.msg
-createTable 0
-client 1
-srsName NAD83_SRS_1
-commitScope 10000
-idcolumn id
-idColumnIsIdentity 1
```

Melhorando o Desempenho da Consulta sobre Dados Espaciais em um Ambiente Particionado

Para alcançar desempenho ideal de consultas em um ambiente de banco de dados particionado, as linhas da tabela que são necessárias para satisfazer a condição de junção devem ser co-localizadas. Isto é, essas junções devem usar linhas de tabela que existam em um nó sem precisar acessar linhas a partir de um tabela ou partes de uma tabela, em um outro nó.

Sobre Esta Tarefa

Junções espaciais são frequentemente utilizadas em aplicativos que localizam clientes dentro de polígonos específicos ou determinam o risco de inundação de clientes. Aqui está um exemplo de uma consulta para determinar risco de inundação:

```
Select
  c.name,
  c.address,
  f.risk
from customers as c,
         floodpoly as f
where db2gse.st_within(c.location, f.polygeom) = 1
```

Nesse exemplo, a tabela `clientes` é bastante grande e distribuída ao longo de múltiplas partições. As informações do polígono de inundação são pequenas. Para implementar a consulta eficientemente, certifique-se de que toda a tabela de polígonos de inundação esteja disponível em cada partição que contiver dados do cliente.

Procedimento

1. Importe os dados do polígono em uma partição simples.

2. Crie uma tabela de consulta materializada (MQT) que seja replicada ao longo das partições que contêm os dados do cliente. Por exemplo, depois que a tabela `floodpoly` tiver sido criada e carregada, você pode criar a MQT replicada com uma instrução semelhante a:

```
CREATE TABLE floodpoly
AS ( SELECT * FROM  floodpoly)
DATA INITIALLY DEFERRED
REFRESH DEFERRED
ENABLE QUERY OPTIMIZATION
MAINTAINED BY SYSTEM
DISTRIBUTE BY REPLICATION IN nodestbs

REFRESH TABLE floodpoly;
```

   A MQT replicada com uma coluna espacial pode ser mantida pelo usuário ou pelo sistema, mas ela deve usar a opção REFRESH DEFERRED e não a opção REFRESH IMMEDIATE.

3. Para fazer o compilador de consulta do DB2 escolher a MQT ao invés da tabela base, é possível configurar os seguintes parâmetros do DB2:

```
UPDATE DB CFG USING DFT_REFRESH_AGE ANY;
UPDATE DB CFG USING DFT_MTTB_TYPES ALL;
```

   Para obter informações adicionais sobre esses parâmetros, consulte "Comando UPDATE DATABASE CONFIGURATION " no *Command Reference*.

O que Fazer Depois

Use as ferramentas de explicação do DB2 para determinar quão eficientemente a consulta será executada.

Capítulo 11. Utilizando Índices e Visualizações para Acessar Dados Espaciais

Use índices e visualizações para consultar colunas espaciais.

Antes de consultar colunas espaciais, você deve aprender os detalhes da criação de índices e visualizações para acessar colunas espaciais. Para criar tais índices, você deve entender a natureza dos índices usados pelo Spatial Extender para facilitar o acesso a dados espaciais.

Índices de Grades Espaciais

Os índices melhoram o desempenho da consulta de aplicativos, principalmente quando a tabela ou tabelas consultadas contêm muitas linhas. Se você criar índices apropriados para o otimizador de consulta escolher para executar sua consulta, poderá reduzir significativamente o número de linhas a serem processadas.

O DB2 Spatial Extender fornece um índice de grade que é otimizado para dados bidimensionais. O índice é criado nas dimensões X e Y de uma geometria.

Os seguintes aspectos de um índice de grade são úteis para entender:

- A geração do índice
- A utilização de funções espaciais em uma consulta
- Como uma consulta utiliza um índice de grade espacial

Geração de Índices de Grade Espaciais

O Spatial Extender gera um índice de grade espacial utilizando o MBR (Minimum Bounding Rectangle) de uma geometria.

Para a maioria das geometrias, o MBR é um retângulo que engloba a geometria.

Um índice de grade espacial divide uma região em grades quadradas lógicas com um tamanho fixo que você especifica quando cria o índice. O índice espacial é construído em um coluna espacial por meio da criação de uma ou mais entradas para as interseções de cada MBR da geometria com as células da grade. Uma entrada de índice consiste no identificador da célula da grade, no MBR da geometria e no identificador interno da linha que contém a geometria.

Você pode definir até três níveis de índice espacial (níveis de grade). A utilização de vários níveis de grade é útil porque permite otimizar o índice para diferentes tamanhos de dados espaciais.

Se uma geometria cruzar quatro ou mais células da grade, a geometria será promovida para o próximo nível mais alto. Em geral, as geometrias maiores serão indexadas nos níveis mais altos. Se uma geometria cruzar 10 ou mais células de grade no maior tamanho da grade, será usado um nível de índice de estouro integrado. Esse nível de estouro impede a geração de excessivas entradas de índices. Para obter melhor desempenho, defina os tamanhos de grades para evitar a utilização desse nível de estouro.

Por exemplo, se existirem vários níveis de grades, o algoritmo de indexação tentará utilizar o menor nível de grade possível para fornecer a melhor resolução para os dados indexados. Quando uma geometria cruza mais de quatro células de grade em determinado nível, ela é promovida para o próximo nível maior (desde que exista outro nível). Portanto, um índice espacial que tem os três níveis de grade de 10,0, 100,0 e 1000,0 primeiro cruzará cada geometria com a grade de nível 10,0. Se uma geometria cruzar com mais de quatro células da grade de tamanho 10,0, ela será promovida e cruzada com a grade de nível 100.0. Se mais de quatro interseções resultarem no nível 100.0, a geometria será promovida para o nível 1000,0. Se mais de 10 interseções resultarem no nível 1000,0, a geometria será indexada no nível do estouro.

Utilização de Funções Espaciais em uma Consulta

Usar funções espaciais na cláusula WHERE faz o otimizador do DB2 considerar índices de grade espacial para o plano de acesso.

As funções espaciais que têm este efeito no otimizador do DB2 são:

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

Como uma consulta utiliza um índice de grade espacial

Quando o otimizador de consulta escolhe um índice de grade espacial, a execução da consulta utiliza um processo de filtragem de várias etapas.

O processo de filtragem inclui as seguintes etapas:

1. Determine as células da grade que cruzam a janela de consulta. A *janela de consulta* é a geometria de seu interesse e que você especifica como o segundo parâmetro em uma função espacial (consulte os exemplos a seguir).
2. Varra o índice em busca de entradas que possuem identificadores de células de grade correspondentes.
3. Compare os valores MBR de geometria nas entradas de índice com a janela de consulta e descarte os valores que estão fora da janela de consulta.
4. Execute análise adicional, se necessário. O conjunto de geometrias candidato das etapas anteriores pode passar por análise adicional para determinar se elas atendem a função espacial (ST_Contains, ST_Distance, etc). A função espacial EnvelopesIntersect omite esta etapa e, geralmente, possui o melhor desempenho.

Os exemplos de consultas espaciais a seguir possuem um índice de grade espacial na coluna C.GEOMETRY:

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

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

No primeiro exemplo, os quatro valores de coordenadas definem a janela de consulta. Estes valores de coordenadas especificam os cantos inferior esquerdo e superior direito (42.0 –73.0 e 43.0 –72.0) de um retângulo.

No segundo exemplo, o Spatial Extender calcula o MBR da geometria especificada pela variável do host :geometry2 e utiliza-o como a janela de consulta.

Quando criar um índice de grade espacial, você deve especificar tamanhos de grades apropriados para os tamanhos de janelas de consulta mais comuns que seu aplicativo espacial provavelmente utilizará. Se um tamanho de grade for maior, as entradas de índice para geometrias que estão fora da janela de consulta deverão ser varridas porque residem nas células de grade que cruzam a janela de consulta e essas varreduras extras reduzem o desempenho. No entanto, um tamanho de grade menor pode gerar mais entradas de índice para cada geometria e entradas de índice adicionais devem ser varridas, o que também reduz o desempenho da consulta.

O DB2 Spatial Extender fornece um utilitário Index Advisor que analisa os dados da coluna espacial e sugere tamanhos de grades apropriados para tamanhos de janela de consulta típica.

Considerações para o Número de Níveis de Índice e Tamanhos de Grade

Utilize o Index Advisor para determinar tamanhos de grade apropriados para os índices de grade espaciais, pois esta é a melhor forma de ajustar os índices e tornar as consultas espaciais mais eficientes.

Número de Níveis de Grade

Você pode ter até três níveis de grade.

Para cada nível de grade em um índice de grade espacial, é executada uma procura de índice separada durante uma consulta espacial. Portanto, se houver mais níveis de grade, a consulta será menos eficiente.

Se os valores na coluna espacial tiverem quase o mesmo tamanho relativo, utilize um único nível de grade. No entanto, uma coluna espacial típica não contém geometrias do mesmo tamanho relativo, mas as geometrias em uma coluna espacial podem ser agrupadas de acordo com o tamanho. Você deve corresponder seus níveis de grade com estes agrupamentos de geometrias.

Por exemplo, suponha que você tenha uma tabela de terrenos de uma região com uma coluna espacial que contenha agrupamentos de pequenos terrenos urbanos cercados por grandes terrenos rurais. Como os tamanhos dos terrenos podem ser colocados em dois grupos (pequenos terrenos urbanos e grandes terrenos rurais), você pode especificar dois níveis de grade para o índice de grade espacial.

Tamanhos de Células de Grade

A regra geral é reduzir os tamanhos de grades o máximo possível para obter a melhor resolução ao reduzir o número de entradas de índice.

- Um valor pequeno deve ser utilizado para o menor tamanho da grade para otimizar todo o índice de geometrias pequenas na coluna. Isto evita a sobrecarga de avaliar geometrias que não estão na área de pesquisa. No entanto, o menor tamanho da grade também gera o maior número de entradas de índices. Consequentemente, o número de entradas de índices processadas no momento da consulta aumenta, conforme a quantidade de armazenamento necessário para o índice. Estes fatores reduzem o desempenho geral.
- Ao utilizar tamanhos maiores de grades, o índice pode ser otimizado para geometrias maiores. Os maiores tamanhos de grades geram menos entradas de índices para grandes geometrias do que o menor tamanho de grade. Consequentemente, os requisitos de armazenamento para o índice são reduzidos, aumentando o desempenho geral.

A figura a seguir mostra os efeitos de diferentes tamanhos de grades.

A Figura 13 mostra um mapa de lotes de terra, cada lote representado por uma geometria de polígono. O retângulo preto representa uma janela de consulta. Suponha que você deseja localizar todas as geometrias cujo MBR cruze a janela de consulta. A Figura 13 mostra que 28 geometrias (realçadas em rosa) possuem um MBR que cruza a janela de consulta.

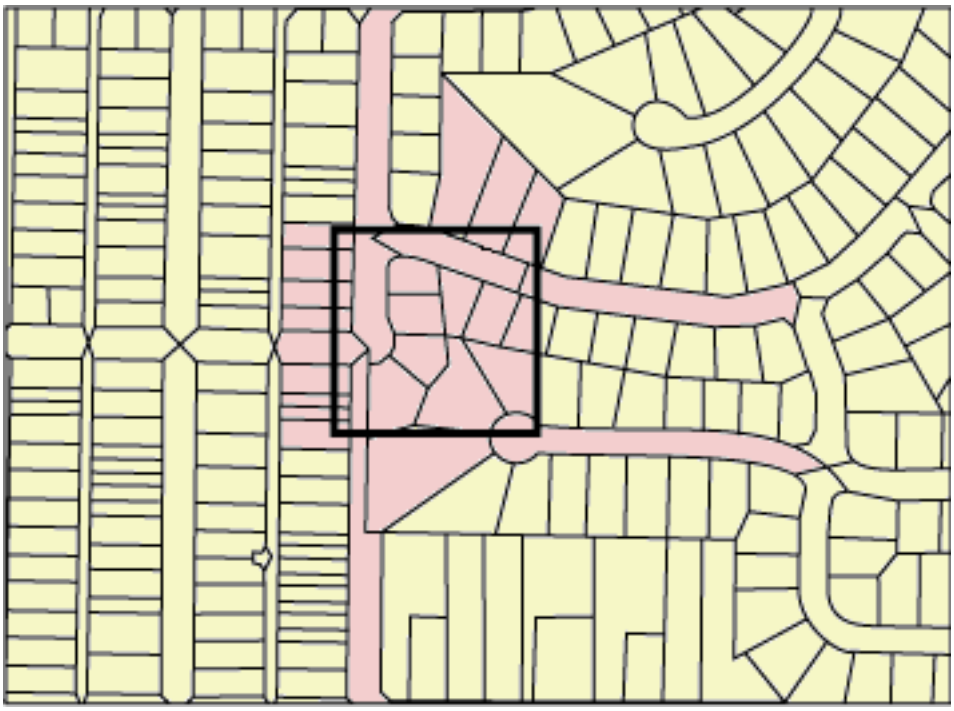


Figura 13. Lotes de terra em uma vizinhança

A Figura 14 na página 77 mostra um tamanho de grade pequeno (25) que fornece um ajuste aproximado da janela de consulta.

- A consulta retorna apenas as 28 geometrias que estão realçadas, mas a consulta deve examinar e descartar três geometrias adicionais cujos MBRs cruzam a janela de consulta.

- Este tamanho de grade pequeno resulta em muitas entradas de índice por geometria. Durante a execução, a consulta acessa todas as entradas de índice para essas 31 geometrias. A Figura 14 mostra 256 células de grade que sobrepõem a janela de consulta. No entanto, a execução da consulta acessa 578 entradas de índice porque muitas geometrias são indexadas com as mesmas células de grade.

Para esta janela de consulta, este tamanho de grade pequeno resulta em um número excessivo de entradas de índice a serem varridas.

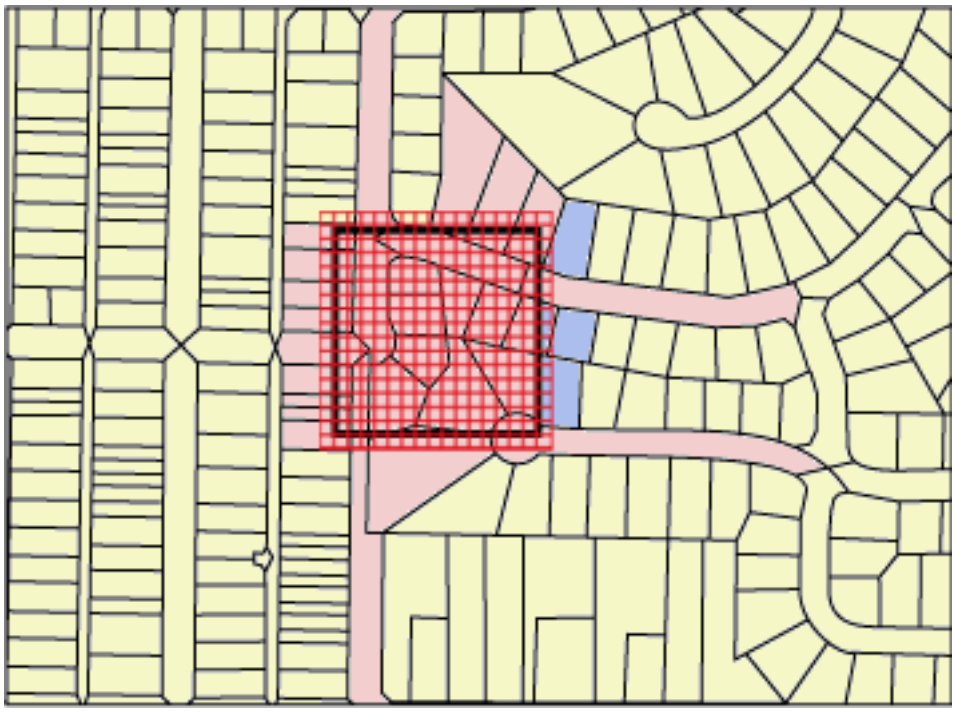


Figura 14. Tamanho de grade pequeno (25) em lotes de terra

A Figura 15 na página 78 mostra um tamanho de grade grande (400) que inclui uma área consideravelmente maior com muito mais geometrias do que a janela de consulta.

- Este tamanho de grade grande resulta em apenas uma entrada de índice por geometria, mas a consulta deve examinar e descartar 59 geometrias adicionais cujos MBRs cruzam a célula de grade.
- Durante a execução, a consulta acessa todas as entradas de índice para as 28 geometrias que cruzam a janela de consulta, além das entradas de índice para as 59 geometrias adicionais, para um total de 112 entradas de índice.

Para esta janela de consulta, este tamanho de grade grande resulta em um número excessivo de geometrias a serem examinadas.

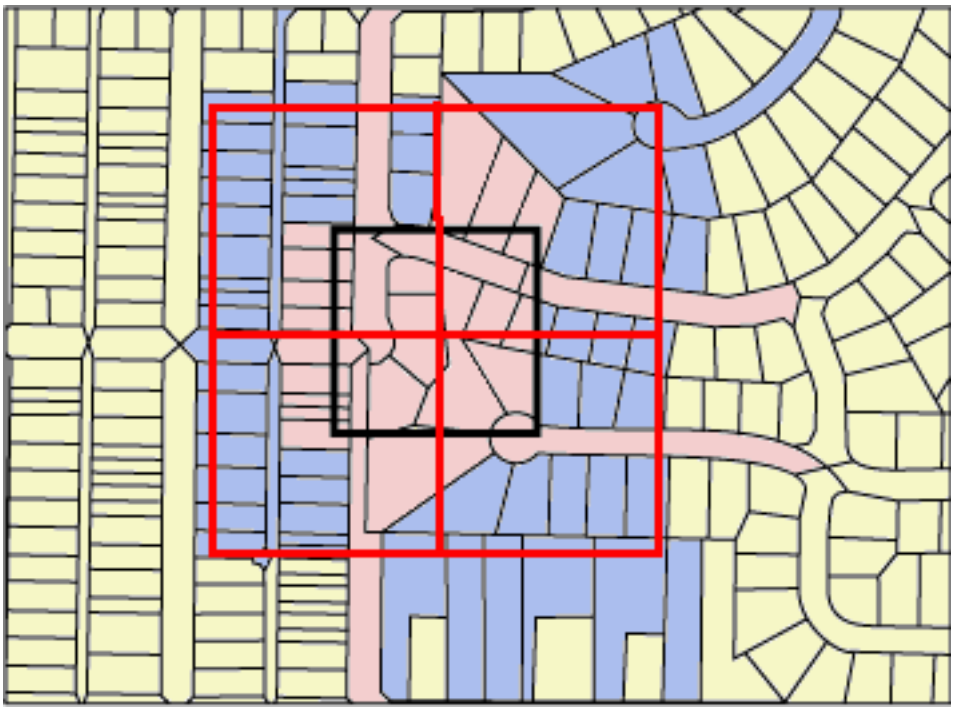


Figura 15. Tamanho de grade grande (400) em lotes de terra

A Figura 16 na página 79 mostra um tamanho de grade médio (100) que fornece um ajuste aproximado da janela de consulta.

- A consulta retorna apenas as 28 geometrias que estão realçadas, mas a consulta deve examinar e descartar cinco geometrias adicionais cujos MBRs cruzam a janela de consulta.
- Durante a execução, a consulta acessa todas as entradas de índice para as 28 geometrias que cruzam a janela de consulta, além das entradas de índice para as 5 geometrias adicionais, para um total de 91 entradas de índice.

Para esta janela de consulta, esta tamanho médio de grade é o melhor porque ele resulta em muito menos entradas de índice do que o tamanho de grade pequeno e a consulta examina menos geometrias adicionais do que o tamanho de grade grande.

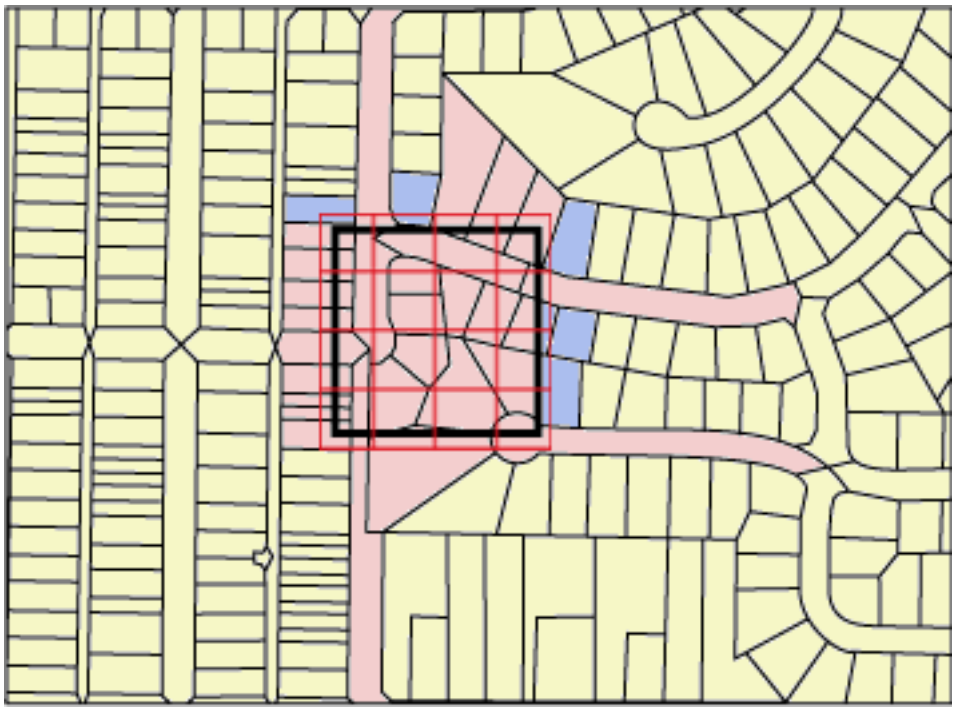


Figura 16. Tamanho de grade médio (100) em lotes de terra

Criando Índices de Grades Espaciais

Crie índices de grade espaciais para definir índices de grade bidimensional em colunas espaciais para ajudar a otimizar consultas espaciais.

Antes de Iniciar

Antes de criar um índice de grade espacial:

- Seu ID de usuário deve reter as autorizações necessárias para a instrução CREATE INDEX SQL.
- Você deve conhecer os valores que deseja especificar para o nome completo do índice de grade espacial e os três tamanhos de grades que o índice utilizará.

Recomendações:

- Antes de criar um índice de grade espacial em uma coluna, utilize o Index Advisor para determinar os parâmetros para o índice. O Index Advisor pode analisar os dados da coluna espacial e sugerir tamanhos de grades apropriados para seu índice de grade espacial.
- Se você planeja fazer um carregamento inicial de dados na coluna, deverá criar o índice de grade espacial depois de concluir o processo de carregamento. Dessa forma, você pode escolher os melhores tamanhos de células de grade que estão baseados nas características dos dados, utilizando o Index Advisor. Além disso, carregar os dados antes de criar o índice melhora o desempenho do processo de carregamento, pois o índice de grade espacial não precisará ser mantido durante o processo de carregamento.

Sobre Esta Tarefa

É possível criar índices de grades espaciais para melhorar o desempenho de consultas nas colunas espaciais. Quando cria um índice de grade espacial, você deve fornecer as seguintes informações:

- Um nome
- O nome da coluna espacial na qual ele será definido
- A combinação dos três tamanhos de grades ajuda a otimizar o desempenho reduzindo o número total de entradas de índice e o número de entradas de índice que precisam ser varridas para atender uma consulta.

Esta tarefa descreve as etapas sobre como criar índices de grades espaciais usando a instrução CREATE INDEX SQL. É possível também criar índices de grades espaciais usando uma ferramenta GIS que trabalha com o DB2 Spatial Extender. Para obter informações sobre como utilizar uma ferramenta GIS para criar um índice de grade espacial, consulte a documentação que acompanha essa ferramenta.

Procedimento

Para criar índices de grades espaciais:

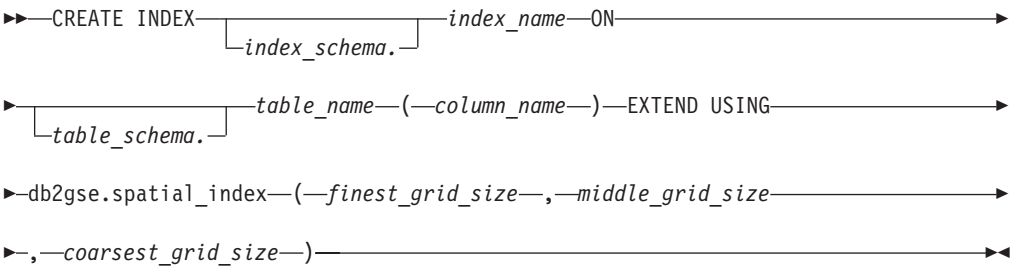
Emita o comando CREATE INDEX no processador de linha de comandos do DB2 especificando a extensão do índice de grade db2gse.spatial_index com a cláusula EXTEND USING. O exemplo a seguir mostra como criar o índice de grade espacial TERRIDX para a tabela BRANCHES que possui uma coluna espacial TERRITORY.

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

Instrução CREATE INDEX para um Índice de Grade Espacial

Utilize a instrução CREATE INDEX com a cláusula EXTEND USING para criar um índice de grade espacial.

Sintaxe



Parâmetros

index_schema.
Nome do esquema ao qual deve pertencer o índice que está sendo criado. Se não for especificado um nome, o sistema de banco de dados do DB2 utilizará o nome do esquema que está armazenado no registro especial CURRENT SCHEMA.

index_name
Nome não qualificado do índice da grade que está sendo criado.

table_schema.
Nome do esquema ao qual pertence a tabela que contém *column_name*. Se não for especificado um nome, o DB2 utilizará o nome do esquema que está armazenado no registro especial CURRENT SCHEMA.

table_name
Nome não qualificado da tabela que contém *column_name*.

column_name
Nome da coluna espacial na qual é criado o índice de grade espacial.

finest_grid_size, middle_grid_size, coarsest_grid_size
Tamanhos de grades para o índice de grade espacial. Estes parâmetros devem atender as seguintes condições:

- *finest_grid_size* deve ser maior do que 0.
- *middle_grid_size* deve ser maior que *finest_grid_size* ou deve ser 0.
- *coarsest_grid_size* deve ser maior que *middle_grid_size* ou deve ser 0.

Quando você cria o índice de grade espacial usando a instrução CREATE INDEX, a validade dos tamanhos de grades são verificadas quando a primeira geometria é indexada. Portanto, se os tamanhos de grade especificados não atenderem as condições de seus valores, ocorrerá uma condição de erro nos momentos descritos nestas situações:

- Se todas as geometrias na coluna espacial forem nulas, o Spatial Extender criará com êxito o índice, sem verificar a validade dos tamanhos de grades. O Spatial Extender valida os tamanhos de grades quando você insere ou atualiza uma geometria não nula nessa coluna espacial. Se os tamanhos de grades especificados não forem válidos, ocorrerá um erro quando você inserir ou atualizar a geometria não nula.
- Se existirem geometrias não nulas na coluna espacial enquanto você cria o índice, o Spatial Extender validará os tamanhos de grades neste momento. Se os tamanhos de grades especificados não forem válidos, ocorrerá um erro imediatamente e o índice de grade espacial não será criado.

Exemplo

A instrução CREATE INDEX de exemplo a seguir cria o índice de grade espacial TERRIDX na coluna espacial TERRITORY na tabela BRANCHES:

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

Ajustando Índices de Grade Espaciais com o Index Advisor

O DB2 Spatial Extender fornece o utilitário Orientador de Índice para ajudá-lo a otimizar o acesso a dados espaciais.

Use o Orientador de Índice para:

- Determinar os tamanhos de grades apropriados para seus índices de grade espaciais.

  O Index Advisor analisa as geometrias em uma coluna espacial e recomenda os tamanhos de grades adequados para seu índice de grade espacial.
- Analisar um índice de grade existente.

O Index Advisor pode coletar e exibir estatísticas a partir das quais você pode determinar como os tamanhos de células de grade atuais facilitam a recuperação dos dados espaciais.

Determinando Tamanhos de Grade para um Índice de Grade Espacial

Antes de criar um índice de grade espacial, use o Orientador de Índice para determinar os tamanhos da grade apropriados.

Antes de Iniciar

- Seu ID do usuário deve conter o privilégio SELECT nesta tabela.
- Se sua tabela tiver mais de um milhão de linhas, talvez você queira utilizar a cláusula ANALYZE para analisar um subconjunto das linhas para ter um tempo de processamento razoável.

Procedimento

Para determinar os tamanhos de grades apropriados para um índice de grade espacial:

1. Utilize o Index Advisor para determinar um tamanho de célula de grade recomendado para o índice que você deseja criar.
   a. Digite o comando que chama o Index Advisor com a palavra-chave ADVISE para solicitar tamanhos de células de grade. Por exemplo, para chamar o Index Advisor para a coluna SHAPE na tabela COUNTIES, digite:

      ```
      gseidx CONNECT TO mydb USER userID USING password GET GEOMETRY
      STATISTICS FOR COLUMN userID.counties(shape) ADVISE
      ```

      Restrição: Se você inserir este comando **gseidx** a partir de um prompt do sistema operacional, deverá digitar o comando inteiro em uma única linha. Como alternativa, você pode executar comandos **gseidx** a partir de um arquivo CLP, que permite dividir o comando em várias linhas.

      O Index Advisor retorna os tamanhos de células de grade recomendados. Por exemplo, o comando **gseidx** com a palavra-chave **ADVISE** mostrada anteriormente retorna os seguintes tamanhos de células recomendados para a coluna SHAPE:

      ```
      Tamanho da Janela de Consulta  Tamanhos de Grades Sugeridos    Custo
      -----------------  -------------------------       ----
             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. Escolha um tamanho de janela de consulta apropriado a partir da saída **gseidx**, com base na largura das coordenadas exibidas em sua tela.

      Neste exemplo, os valores de latitude e longitude em graus decimais representam as coordenadas. Se sua exibição de mapa típica tiver uma largura de aproximadamente 0,5 graus (aproximadamente 55 quilômetros), vá para a linha que possui o valor 0,5 na coluna Tamanho da Janela de Consulta. Esta linha sugeriu tamanhos de grade de 1.4, 3.5 e 14.0.
2. Crie o índice com os tamanhos de grade sugeridos. Por exemplo, na etapa anterior, você pode executar a seguinte instrução SQL:

```
CREATE INDEX counties_shape_idx ON userID.counties(shape)
EXTEND USING DB2GSE.SPATIAL_INDEX(1.4,3.5,14.0);
```

Analisando Estatísticas de Índice de Grade Espacial

A análise de estatísticas sobre um índice de grade espacial existente pode informar se o índice é eficiente ou se ele deve ser substituído por um índice mais eficiente.

Antes de Iniciar

Antes de analisar os dados que deseja indexar:

- Seu ID do usuário deve conter o privilégio SELECT nesta tabela.
- Se sua tabela tiver mais de um milhão de linhas, talvez você queira utilizar a cláusula ANALYZE para analisar um subconjunto das linhas para ter um tempo de processamento razoável. Você deve ter um espaço de tabelas USER TEMPORARY disponível para utilizar a cláusula ANALYZE. Configure o tamanho da página deste espaço de tabelas para pelo menos 8 KB e certifique-se de que tenha privilégios USE para ele. Por exemplo, as seguintes instruções DDL criam um conjunto de buffers com o mesmo tamanho de página que o espaço de tabelas temporário do usuário e concedem o privilégio USE a qualquer pessoa:

```
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;
```

Sobre Esta Tarefa

Use o Index Advisor para obter estatísticas sobre um índice de grade espacial existente. Analise essas estatísticas e determine se é necessário substituir algum índice.

Dica: Igualmente importante para ajustar seu índice é verificar se ele está sendo utilizado por suas consultas. Para determinar se um índice espacial está sendo usado, execute uma ferramenta de linha de comandos como **db2exfmt** em sua consulta. Na seção "Plano de Acesso" da saída de explicação, se você vir um operador EISCAN e o nome de seu índice espacial, a consulta utilizará seu índice.

Procedimento

Para analisar estatísticas sobre um índice de grade espacial existente e determinar se ele deve ser substituído por um índice mais eficiente:

Obtenha estatísticas sobre um índice de grade espacial e, se necessário, para substituir o índice:

1. Deixe o Index Advisor coletar estatísticas com base nos tamanhos de células de grade do índice existente. Você pode solicitar estatísticas sobre um subconjunto de dados indexados ou sobre todos os dados.
   - Para obter estatísticas de dados indexados em um subconjunto de linhas, insira o comando **gseidx** e especifique a palavra-chave **ANALYZE** e seus parâmetros, além da cláusula de índice existente e da palavra-chave **DETAIL**. Você pode especificar o número ou a porcentagem de linhas que o Index Advisor deve analisar para obter estatísticas. Por exemplo, para obter estatísticas sobre um subconjunto dos dados indexados pelo índice COUNTIES_SHAPE_IDX, digite:

```
gseidx CONNECT TO mydb USER userID USING password GET GEOMETRY
STATISTICS FOR INDEX userID.counties_shape_idx DETAIL ANALYZE 25 PERCENT
 ADVISE
```

- Para obter estatísticas sobre todos os dados indexados, digite o comando **gseidx** e especifique sua cláusula de índice existente. Inclua a palavra-chave **DETAIL**. Por exemplo, para chamar o Index Advisor para o índice COUNTIES_SHAPE_IDX, digite:

```
gseidx CONNECT TO mydb USER userID USING password GET GEOMETRY
STATISTICS FOR INDEX userID.counties_shape_idx DETAIL SHOW HISTOGRAM ADVISE
```

O Index Advisor retorna estatísticas, um histograma dos dados e tamanhos de células recomendados para o índice existente. Por exemplo, o comando **gseidx** anterior para todos os dados indexados por COUNTIES_SHAPE_IDX retorna as seguintes estatísticas:

```
Nível de Grade 1
------------

Tamanho da Grade              : 0.5
Número de Geometrias          : 2936
Número de Entradas de Índice  : 12197

Número de Células de Grade Ocupadas       : 2922
Proporção Entrada de Índice/Geometria     : 4.154292
Proporção Geometria/Célula da Grade       : 1.004791
Número máximo de Geometrias por Célula da Grade : 14
Número mínimo de Geometrias por Células da Grade: 1

Entradas de Índice : 1     2     3      4      10
---------------  ------ ------ ------ ------ ------
Absoluto      :  86     564    72     1519   695
Porcentagem (%): 2.93   19.21  2.45   51.74  23.67



Nível de Grade 2
------------

Tamanho da Grade              : 0.0
Não existem geometrias indexadas neste nível.


Nível de Grade 3
------------

Tamanho da Grade              : 0.0
Não existem geometrias indexadas neste nível.


Nível de Grade X
------------

Número de Geometrias          : 205
Número de Entradas de Índice  : 205
```

2. Determine como os tamanhos de células de grade do índice existente facilitarão a recuperação. Avalie as estatísticas retornadas na etapa anterior.

   Dica:

   - A estatística "Proporção Entrada de Índice/Geometria" deve ser um valor no intervalo de 1 a 4, preferencialmente valores mais próximos de 1.
   - O número de entradas de índice por geometria deve ser menor que 10 no maior tamanho de grade para evitar o nível de estouro.

A aparência da seção "Nível de Grade X" na saída do Index Advisor indica que existe um nível de estouro.

As estatísticas de índice obtidas na etapa anterior para COUNTIES_SHAPE_IDX indicam que os tamanhos de grade (0.5, 0, 0) não são apropriados para os dados nesta coluna porque:

- Para o Nível de Grade 1, o valor `4,154292` para a "Proporção Entrada de Índice/Geometria" é maior que a diretriz de 4.

  A linha de "Entradas de Índice" possui os valores 1, 2, 3, 4 e 10, que indica o número de entradas de índice por geometria. Os valores "Absolutos" abaixo de cada coluna de "Entradas de Índice" indicam o número de geometrias que possuem um número específico de entradas de índice. Por exemplo, a saída na etapa anterior mostra 1519 geometrias que possuem 4 entradas de índice. O valor "Absoluto" para 10 entradas de índice é 695, que indica que 695 geometrias possuem entre 5 e 10 entradas de índice.
- A aparência da seção "Nível de Grade X" indica que existe um nível de índice de estouro. As estatísticas mostram que 205 geometrias possuem mais de 10 entradas de índice cada.

3. Se as estatísticas não forem satisfatórias, consulte a seção Histograma e as linhas apropriadas nas colunas Tamanho da Janela de Consulta e Tamanhos de Grades Sugeridos na saída do Index Advisor.
   a. Localize o tamanho do MBR com o maior número de geometrias. A seção "Histograma" lista os tamanhos de MBR e o número de geometrias que possuem esse tamanho de MBR. No histograma de amostra a seguir, o maior número de geometrias (437) está no tamanho de MBR 0.5.

```
Histograma:
----------
    Tamanho de MBR       Contagem de Geometrias
    -------------------- --------------------
                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. Vá para a linha Tamanho da Janela de Consulta com o valor 0.5 para obter os tamanhos de grades sugeridos (1.4, 3.5, 14.0).

```
Tamanho da Janela de Consulta  Tamanhos de Grades Sugeridos    Custo
-----------------  -------------------------       ----

       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. Verifique se os tamanhos recomendados atendem as diretrizes na etapa 2. Execute o comando **gseidx** com os tamanhos de grades sugeridos:

```
gseidx CONNECT TO mydb USER userID USING password GET GEOMETRY
 STATISTICS FOR COLUMN userID.counties(shape) USING GRID SIZES (1.4, 3.5, 14.0)

Nível de Grade 1
------------

Tamanho da Grade             : 1.4
Número de Geometrias         : 3065
Número de Entradas de Índice  : 5951

Número de Células de Grade Ocupadas             : 513
Proporção Entrada de Índice/Geometria           : 1.941599
Proporção Geometria/Célula da Grade             : 5.974659
Número máximo de Geometrias por Célula da Grade : 42
Número mínimo de Geometrias por Células da Grade: 1

Entradas de Índice : 1     2     3      4      10
---------------  ------ ------ ------ ------ ------
Absoluto       : 1180   1377   15     493    0
Porcentagem (%): 38.50  44.93  0.49   16.08  0.00


Nível de Grade 2
------------

Tamanho da Grade             : 3.5
Número de Geometrias         : 61
Número de Entradas de Índice  : 143

Número de Células de Grade Ocupadas             : 56
Proporção Entrada de Índice/Geometria           : 2.344262
Proporção Geometria/Célula da Grade             : 1.089286
Número máximo de Geometrias por Célula da Grade : 10
Número mínimo de Geometrias por Células da Grade: 1

Entradas de Índice : 1     2     3      4      10
---------------  ------ ------ ------ ------ ------
Absoluto       : 15     28     0      18     0
```

```
Porcentagem (%): 24.59  45.90  0.00   29.51  0.00


Nível de Grade 3
------------

Tamanho da Grade               : 14.0
Número de Geometrias           : 15
Número de Entradas de Índice   : 28

Número de Células de Grade Ocupadas             : 9
Proporção Entrada de Índice/Geometria           : 1.866667
Proporção Geometria/Célula da Grade             : 1.666667
Número máximo de Geometrias por Célula da Grade : 10
Número mínimo de Geometrias por Células da Grade: 1

Entradas de Índice : 1     2     3      4      10
---------------  ------ ------ ------ ------ ------
Absoluto      :  7      5      1      2      0
Porcentagem (%): 46.67  33.33  6.67   13.33  0.00
```

As estatísticas agora mostram valores nas diretrizes:

- Os valores da "Proporção Entrada de Índice/Geometria" são 1.941599 para o Nível de Grade 1, 2.344262 para o Nível de Grade 2 e 1.866667 para o Nível de Grade 3. Estes valores estão no intervalo de valores de diretrizes de 1 a 4.
- A ausência da seção "Nível de Grade X" indica que não existe nenhuma entrada de índice no nível de estouro.

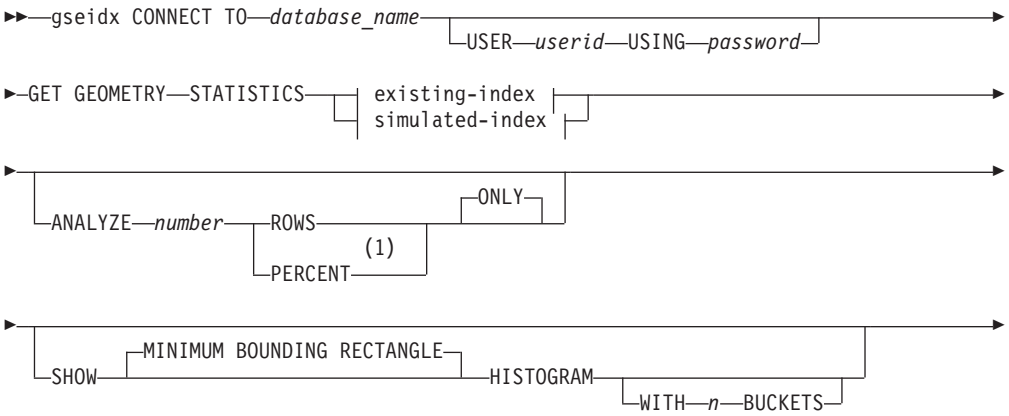
5. Elimine o índice existente e substitua-o por um índice que especifique os tamanhos de grades recomendados. Para a amostra na etapa anterior, execute as seguintes instruções DDL:

```
DROP INDEX userID.counties_shape_idx;
CREATE INDEX counties_shape_idx ON userID.counties(shape) EXTEND USING
  DB2GSE.SPATIAL_INDEX(1.4,3.5,14.0);
```

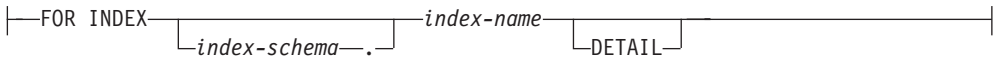
Comando gseidx

Utilize o comando **gseidx** para chamar o Index Advisor para índices de grade espaciais.

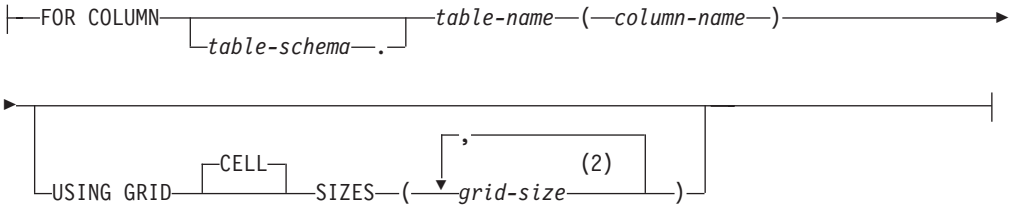
Sintaxe



existing-index:



simulated-index:



Notas:

1 Em vez da palavra-chave PERCENT, você pode especificar um sinal de porcentagem (%).

2 Você pode especificar tamanhos de células para um, dois ou três níveis de grade.

Parâmetros

database_name
O nome do banco de dados no qual a tabela espacial reside.

userid O ID do usuário que possui autoridade DATAACCESS no banco de dados no qual o índice ou a tabela reside ou autoridade SELECT na tabela. Se você efetuar logon no ambiente de comandos do DB2 com o ID do usuário do proprietário do banco de dados, não será necessário especificar *userid* e *password* no comando **gseidx**.

password
Senha do ID do usuário.

existing-index

Refere-se a um índice existente do qual serão coletadas estatísticas.

index-schema
Nome do esquema que inclui o índice existente.

index-name
Nome não qualificado do índice existente.

DETAIL
Mostra as seguintes informações sobre cada nível de grade:

- O tamanho das células de grade
- O número de geometrias indexadas
- O número de entradas do índice
- O número de células de grade que contêm geometrias
- O número médio de entradas de índice por geometria

- O número médio de geometrias por célula de grade
- O número de geometrias na célula que contém a maior quantidade de geometrias
- O número de geometrias na célula que contém a menor quantidade de geometrias

simulated-index

Refere-se a uma coluna de tabela e a um índice simulado para esta coluna.

table-schema
Nome do esquema que inclui a tabela com a coluna à qual se destina o índice simulado.

table-name
Nome não qualificado da tabela com a coluna à qual se destina o índice simulado.

column-name
Nome não qualificado da coluna da tabela à qual se destina o índice simulado.

grid-size
Tamanhos de células em cada nível de grade (nível mais fino, nível médio e nível mais espesso) de um índice simulado. Você deve especificar um tamanho de célula para pelo menos um nível. Se não desejar incluir um nível, não especifique um tamanho de célula de grade para ele ou especifique um tamanho de célula de grade zero (0.0) para ele.

Quando especificar o parâmetro *grid-size*, o Index Advisor retornará os mesmos tipos de estatísticas que ele retorna quando você inclui a palavra-chave DETAIL na cláusula existing-index.

ANALYZE number ROWS | PERCENT ONLY
Especifique a quantidade aproximada ou a porcentagem aproximada das linhas usadas para reunir estatísticas sobre dados. Se sua tabela tiver mais de um milhão de linhas, use a cláusula ANALYZE para reunir estatísticas para um subconjunto de dados para que seja possível ter um tempo de processamento razoável.

SHOW MINIMUM BOUNDING RECTANGLE HISTOGRAM
Exibe um gráfico que mostra os tamanhos dos MBRs (Minimum Bounding Rectangles) das geometrias e o número de geometrias cujos MBRs têm o mesmo tamanho.

WITH n BUCKETS
Especifica o número de agrupamentos para os MBRs de todas as geometrias analisadas. Os MBRs pequenos são agrupados com outras geometrias pequenas. Os MBRs grandes são agrupados com outras geometrias grandes.

Se você não especificar este parâmetro ou especificar 0 reservatórios, o Index Advisor exibirá tamanhos de reservatórios logarítmicos. Por exemplo, o tamanho do MBR pode ser de valores logarítmicos como 1.0, 2.0, 3.0,... 10.0, 20.0, 30.0,... 100.0, 200.0, 300.0,...

Se você especificar um número de reservatórios maior que 0, o Index Advisor exibirá valores de mesmo tamanho. Por exemplo, os

tamanhos de MBR podem ter valores de mesmo tamanho como 8.0, 16.0, 24.0,... 320.0, 328.0, 334.0.

O padrão é utilizar reservatórios de tamanhos logarítmicos.

ADVISE GRID CELL SIZES
Calcula tamanhos de células de grade próximos do ideal.

Nota de Uso

Se você inserir o comando **gseidx** a partir de um prompt do sistema operacional, será necessário digitar todo o comando em uma única linha.

Exemplo

O exemplo a seguir é um pedido para retornar informações detalhadas sobre um índice de grade existente cujo nome é COUNTIES_SHAPE_IDX e para sugerir tamanhos de índice de grade apropriados:

```
gseidx CONNECT TO mydb USER user ID USING password GET GEOMETRY
STATISTICS FOR INDEX userID.counties_shape_idx DETAIL ADVISE
```

Utilizado Visualizações para Acessar Colunas Espaciais

Você pode definir uma exibição que utiliza uma coluna espacial da mesma forma que define exibições no DB2 para outros tipos de dados.

Sobre Esta Tarefa

Isso é particularmente útil quando uma tabela tem várias colunas espaciais, uma vez que muitos aplicativos de visualização podem trabalhar apenas com uma tabela ou visualização com uma única coluna espacial. A definição de visualização pode selecionar a coluna espacial apropriada e outras colunas não espaciais para serem disponibilizadas para o aplicativo.

Capítulo 12. Analisando e Gerando Informações Espaciais

A análise e geração de informações espaciais requerem um entendimento dos ambientes nos quais é possível submeter consultas e as diretrizes sobre como usar funções espaciais juntamente com índices espaciais. Revise os exemplos dos vários tipos de funções espaciais possíveis de chamar em uma consulta para aprender sobre os recursos oferecidos pelo Spatial Extender.

Ambientes para Execução de análise Espacial

É possível executar análise espacial usando SQL e funções espaciais a partir do processador de linha de comandos do DB2 e de programas de Aplicativo em todas as linguagens suportadas pelo DB2.

Exemplo de como Operam as Funções espaciais

O DB2 Spatial Extender fornece funções que executam várias operações em dados espaciais. Estas funções podem ser categorizadas de acordo com o tipo de operação que elas executam.

A Tabela 2 lista essas categorias, junto com exemplos. O texto após a Tabela 2 mostra a codificação destes exemplos.

Tabela 2. Operações e funções espaciais

Categoria de função	Exemplo de operação
Retorna informações sobre geometrias específicas.	Retorna a extensão, em milhas quadradas, da área de vendas da Loja 10.
Faz comparações.	Determina se a localização da residência de um cliente está dentro da área de vendas da Loja 10.
Gera novas geometrias a partir de existentes.	Gera a área de vendas de uma loja a partir de sua localização.
Converte geometrias em e a partir de formatos de troca de dados.	Converte informações do cliente em formato GML em uma geometria para que as informações possam ser incluídas em um banco de dados DB2.

Exemplo 1: Retorna informações sobre geometrias específicas

Neste exemplo, a função ST_Area retorna um valor numérico que representa a área de vendas da loja 10. A função retornará a área nas mesmas unidades que as unidades do sistema de coordenadas que está sendo utilizado para definir a localização da área.

```
SELECT db2gse.ST_Area(sales_area)
FROM   stores
WHERE id = 10
```

O exemplo a seguir mostra a mesma operação que o anterior, mas com ST_Area chamado como um método e retornando a área em unidades de milhas quadradas.

```
SELECT saleas_area..ST_Area('STATUTE MILE')
FROM   stores
WHERE  id = 10
```

Exemplo 2: Faz comparações

Neste exemplo, a função ST_Within compara as coordenadas da geometria representando a residência de um cliente com as coordenadas de uma geometria representando a área de vendas da loja 10. A saída da função definirá se a residência está localizada na área de vendas.

```
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
```

Exemplo 3: Origina novas geometrias de geometrias existentes.

Neste exemplo, a função ST_Buffer gera uma geometria representando uma área de vendas de uma loja a partir de uma geometria representando a localização da loja.

```
UPDATE stores
SET    sales_area = db2gse.ST_Buffer(location, 10, 'KILOMETERS')
WHERE  id = 10
```

O exemplo a seguir mostra a mesma operação que o anterior, mas com ST_Buffer chamado como um método.

```
UPDATE stores
SET    sales_area = location..ST_Buffer(10, 'KILOMETERS')
WHERE  id = 10
```

Exemplo 4: Converte geometrias em e a partir de formatos de troca de dados.

Neste exemplo, as informações do cliente codificadas em GML são convertidas em uma geometria, para que possam ser armazenadas em um banco de dados 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) )
```

Funções que Utilizam Índices para Otimizar Consultas

Um grupo especializado de funções espaciais, chamado *funções de comparação*, pode melhorar o desempenho da consulta explorando índices de grade espacial. Cada uma destas funções compara duas geometrias.

Se os resultados da comparação atenderem alguns critérios, a função retornará um valor 1; se os resultados não atenderem os critérios, a função retornará um valor 0. Se a comparação não puder ser executada, a função poderá retornar um valor nulo.

Por exemplo, a função ST_Overlaps compara duas geometrias que têm a mesma dimensão (por exemplo, duas sequências de linhas ou dois polígonos). Se as geometrias forem parcialmente sobrepostas e se o espaço coberto pela sobreposição tiver a mesma dimensão das geometrias, ST_Overlaps retornará um valor 1.

A Tabela 3 na página 93 mostra quais funções de comparação podem usar um índice de grade espacial:

Tabela 3. Funções de Comparação que Podem Usar um Índice de Grade Espacial

Função de comparação	Pode utilizar índice de grade espacial
EnvelopesIntersect	Sim
ST_Contains	Sim
ST_Crosses	Sim
ST_Distance	Sim
ST_EnvIntersects	Sim
ST_Equals	Sim
ST_Intersects	Sim
ST_MBRIntersects	Sim
ST_Overlaps	Sim
ST_Touches	Sim
ST_Within	Sim

Devido ao tempo e a memória requeridos para executar uma função, tal execução pode envolver um processamento considerável. Além disso, quanto mais complexas forem as geometrias que estão sendo comparadas, mais complexa e mais demorada será a comparação. As funções especializadas listadas anteriormente podem concluir suas operações mais rapidamente se puderem usar um índice espacial para localizar geometrias. Para ativar tal função para utilizar um índice espacial, observe todas as seguintes regras:

- A função deve ser especificada em uma cláusula WHERE. Se for especificada em uma cláusula SELECT, HAVING ou GROUP BY, não será possível utilizar um índice espacial.
- A função deve ser a expressão à esquerda do predicado.
- O operador que é utilizado no predicado que compara o resultado da função com outra expressão deve ser um sinal de igual, com uma exceção: a função ST_Distance deve utilizar o operador menor que.
- A expressão à direita do predicado deve ser a constante 1, exceto quando ST_Distance é a função à esquerda.
- A operação deve envolver uma pesquisa em uma coluna espacial na qual um índice espacial está definido.

Por exemplo:

```
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
```

A Tabela 4 mostra as formas correta e incorreta de criar consultas espaciais para utilizar um índice espacial.

Tabela 4. Demonstração de como as funções espaciais podem estar de acordo e violar regras para utilizar um índice espacial.

Consultas que se referem a funções espaciais	Regras violadas

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Contains(s.sales_zone,`
`ST_Point(-121.8,37.3, 1)) = 1`
	Nenhuma condição é violada neste exemplo.

Tabela 4. Demonstração de como as funções espaciais podem estar de acordo e violar regras para utilizar um índice espacial. (continuação)

Consultas que se referem a funções espaciais	Regras violadas

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Length(s.location) > 10`
	A função espacial ST_Length não compara geometrias e não pode utilizar um índice espacial.

`SELECT *`
`FROM stores AS s`
`WHERE 1=db2gse.ST_Within(s.location,:BayArea)`
	A função deve ser uma expressão à esquerda do predicado.

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Contains(s.sales_zone,`
`ST_Point(-121.8,37.3, 1)) <> 0`
	Comparações de igualdade devem usar o inteiro constante 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`
	Não existe nenhum índice espacial em nenhum dos argumentos da função, portanto, nenhum índice poderá ser utilizado.

Capítulo 13. Escrevendo aplicativos e utilizando o programa de exemplo

Para gravar aplicativos para o Spatial Extender, você deve entender os requisitos e revisar o programa de amostra.

Incluindo o Arquivo de Cabeçalho do DB2 Spatial Extender em Aplicativos Espaciais

O DB2 Spatial Extender fornece um arquivo de cabeçalho que define constantes que podem ser utilizadas com os procedimentos armazenados e funções do DB2 Spatial Extender.

Sobre Esta Tarefa

Recomendação:
Se pretende chamar procedimentos armazenados ou funções do DB2 Spatial Extender a partir de programas C ou C++, inclua esse arquivo de cabeçalho em seus aplicativos espaciais.

Procedimento

Para incluir o arquivo de cabeçalho do DB2 Spatial Extender em aplicativos espaciais:

1. Certifique-se de que seus aplicativos do DB2 Spatial Extender possam utilizar as definições necessárias desse arquivo de cabeçalho.
   a. Inclua o arquivo de cabeçalho do DB2 Spatial Extender em seu programa aplicativo. O arquivo de cabeçalho tem o seguinte nome:

      ```
      db2gse.h
      ```

      O arquivo de cabeçalho está localizado no diretório *db2path*/include, em que *db2path* é o diretório de instalação no qual o sistema de banco de dados DB2 está instalado.
   b. Certifique-se de que o caminho do diretório include esteja especificado em seu `arquivo pronto` com a opção de compilação.
2. Se estiver criando aplicativos Windows de 64 bits em um sistema Windows de 32 bits, altere o parâmetro `DB2_LIBS` no arquivo `samples/extenders/spatial/makefile.nt` para acomodar aplicativos de 64 bits. As mudanças necessárias são destacadas no seguinte exemplo:

   ```
   DB2_LIBS = $(DB2_DIR)\lib\Win64\db2api.lib
   ```

Chamando Procedimentos Armazenados do DB2 Spatial Extender a Partir de um Aplicativo

Se estiver planejando gerar programas aplicativos que chamam qualquer um dos procedimentos armazenados do DB2 Spatial Extender, utilize a instrução SQL CALL e especifique o nome do procedimento armazenado.

Sobre Esta Tarefa

Os procedimentos armazenados do DB2 Spatial Extender são criados quando você ativa o banco de dados para operações espaciais.

Procedimento

Para chamar procedimentos armazenados do DB2 Spatial Extender de um aplicativo:

1. Chame o procedimento armazenado DB2GSE.ST_ENABLE_DB para ativar um banco de dados para operações espaciais.

   Especifique o nome do procedimento armazenado da seguinte forma:

   ```
   CALL DB2GSE!ST_ENABLE_DB
   ```

   O `DB2GSE!` nesta chamada representa o nome da biblioteca do DB2 Spatial Extender. O procedimento ST_ENABLE_DB é o único no qual é necessário incluir um ponto de exclamação na chamada (ou seja, `DB2GSE!`).
2. Chame outros procedimentos armazenados do DB2 Spatial Extender. Especifique o nome do procedimento armazenado no seguinte formato, em que `DB2GSE` é o nome do esquema para todos os procedimentos armazenados do DB2 Spatial Extender e *spatial_procedure_name* é o nome do procedimento armazenado. Não inclua um ponto de exclamação na chamada.

   ```
   CALL DB2GSE.spatial_procedure_name
   ```

   Os procedimentos armazenados do DB2 Spatial Extender são mostrados na tabela a seguir.

Tabela 5. Procedimentos armazenados do DB2 Spatial Extender

Procedimento Armazenado	Descrição
ST_ALTER_COORDSYS	Atualiza um atributo de um sistema de coordenadas no banco de dados.
ST_ALTER_SRS	Atualiza um atributo de um sistema de referência espacial no banco de dados.
ST_CREATE_COORDSYS	Cria um sistema de coordenadas no banco de dados.
ST_CREATE_SRS	Cria um sistema de referência espacial no banco de dados.
ST_DISABLE_AUTOGEOCODING	Especifica que o DB2 Spatial Extender deve parar a sincronização de uma coluna geocodificada com suas colunas de geocodificação associadas.
ST_DISABLE_DB	Remove recursos que permitem que o DB2 Spatial Extender armazene dados espaciais e suporte operações que são executadas nestes dados.
ST_DROP_COORDSYS	Exclui um sistema de coordenadas do banco de dados.
ST_DROP_SRS	Exclui um sistema de referência espacial do banco de dados.
ST_ENABLE_AUTOGEOCODING	Especifica que o DB2 Spatial Extender deve sincronizar uma coluna geocodificada com suas colunas de geocodificação associadas.

Tabela 5. Procedimentos armazenados do DB2 Spatial Extender (continuação)

Procedimento Armazenado	Descrição
ST_ENABLE_DB	Fornece a um banco de dados os recursos necessários para ele armazenar dados espaciais e suportar operações.
ST_EXPORT_SHAPE	Exporta dados selecionados no banco de dados para um arquivo shape.
ST_IMPORT_SHAPE	Importa um arquivo shape para um banco de dados.
ST_REGISTER_GEOCODER	Registra um geocodificador diferente do DB2SE_USA_GEOCODER, que faz parte do produto DB2 Spatial Extender.
ST_REGISTER_SPATIAL_COLUMN	Registra uma coluna espacial e associa a ela um sistema de referência espacial.
ST_REMOVE_GEOCODING_SETUP	Remove todas as informações de configuração de geocodificação da coluna na qual foi geocodificado.
ST_RUN_GEOCODING	Executa um geocodificador no modo em lote.
ST_SETUP_GEOCODING	Associa uma coluna na qual deve ser geocodificado a um geocodificador e configura os valores de parâmetros de geocodificação correspondentes.
ST_UNREGISTER_GEOCODER	Cancela o registro de um geocodificador.
ST_UNREGISTER_SPATIAL_COLUMN	Remove o registro de uma coluna espacial.

Programa de Amostra do DB2 Spatial Extender

O programa de amostra do DB2 Spatial Extender, `runGseDemo`, tem duas finalidades. Você pode utilizar o programa de amostra para se familiarizar com a programação de aplicativos para o DB2 Spatial Extender e pode utilizar o programa para verificar a instalação do DB2 Spatial Extender.

O local do programa runGseDemo varia e depende do sistema operacional em que o DB2 Spatial Extender está instalado.

- No UNIX, é possível localizar o programa runGseDemo no seguinte caminho:
  ```
  $HOME/sqllib/samples/extenders/spatial
  ```
  em que `$HOME` é o diretório pessoal do proprietário da instância.
- No Windows®, você pode localizar o programa runGseDemo no seguinte caminho:
  ```
  c:\Arquivos de Programas\IBM\sqllib\samples\extenders\spatial
  ```
  em que `c:\Arquivos de Programas\IBM\sqllib` é o diretório no qual o DB2 Spatial Extender foi instalado.

O programa de amostra do DB2 Spatial Extender `runGseDemo` facilita a programação de aplicativos. Utilizando este programa de amostra, você poderá ativar um banco de dados para operações espaciais e executará análise espacial em dados nesse banco de dados. Este banco de dados conterá tabelas com informações fictícias sobre clientes e zonas de alagamento. A partir destas informações você

pode efetuar testes com o Spatial Extender e determinar quais clientes estão correndo risco de sofrer danos decorrentes de um alagamento.

Com o programa de amostra, você pode:

- Consultar as etapas geralmente requeridas para criar e manter um banco de dados ativado espacialmente.
- Entender como chamar procedimentos armazenados espaciais a partir de um programa aplicativo.
- Recortar e colar código de amostra em seus próprios aplicativos.

Utilize o seguinte programa de amostra para codificar tarefas para o DB2 Spatial Extender. Por exemplo, suponha que você crie um aplicativo que utiliza a interface do banco de dados para chamar procedimentos armazenados do DB2 Spatial Extender. A partir do programa de amostra, você pode copiar código para personalizar seu aplicativo. Se não estiver familiarizado com as etapas de programação para o DB2 Spatial Extender, você poderá executar o programa de amostra, que mostra cada etapa detalhadamente. Para obter instruções sobre como executar o programa de amostra, consulte a seção "Tarefas Relacionadas" no final deste tópico.

A tabela a seguir descreve cada etapa do programa de amostra. Em cada etapa, você executará uma ação e, em muitos casos, reverterá ou irá desfazer essa ação. Por exemplo, na primeira etapa, você ativará o banco de dados espacial e, em seguida, desativará o banco de dados espacial. Desta forma, você se familiarizará com muitos dos procedimentos armazenados do Spatial Extender.

Tabela 6. Etapas do Programa de Amostra do DB2 Spatial Extender

Etapas do SQL	Ação e Descrição
Ativar ou desativar o banco de dados espacial	• Ative o banco de dados espacial Esta é a primeira etapa necessária para utilizar o DB2 Spatial Extender. Um banco de dados que foi ativado para operações espaciais tem um conjunto de tipos espaciais, um conjunto de funções espaciais, um conjunto de predicados espaciais, novos tipos de índices e um conjunto de tabelas de catálogos espaciais e exibições. • Desative o banco de dados espacial Esta etapa geralmente é executada quando você tiver ativado capacidades espaciais para o banco de dados incorreto ou quando não mais precisar executar operações espaciais neste banco de dados. Ao desativar um banco de dados espacial, você remove o conjunto de tipos espaciais, o conjunto de funções espaciais, o conjunto de predicados espaciais, novos tipos de índices e o conjunto de tabelas de catálogos espaciais e exibições associadas a esse banco de dados. • Ative o banco de dados espacial Igual à ação anterior.

Tabela 6. Etapas do Programa de Amostra do DB2 Spatial Extender (continuação)

Etapas do SQL	Ação e Descrição
Criar ou eliminar um sistema de coordenadas	• Crie um sistema de coordenadas denominado NORTH_AMERICAN Esta etapa cria um novo sistema de coordenadas no banco de dados. • Elimine o sistema de coordenadas denominado NORTH_AMERICAN Esta etapa elimina o sistema de coordenadas NORTH_AMERICAN do banco de dados. • Crie um sistema de coordenadas denominado KY_STATE_PLANE Esta etapa cria um novo sistema de coordenadas, KY_STATE_PLANE, que será utilizado pelo sistema de referência espacial criado na etapa seguinte.
Criar ou eliminar um sistema de referência espacial	• Crie um sistema de referência espacial denominado SRSDEMO1 Esta etapa define um novo SRS (Sistema de Referência Espacial) que é utilizado para interpretar as coordenadas. O SRS inclui dados de geometria em um formato que pode ser armazenado em uma coluna de um banco de dados ativado espacialmente. Depois que o SRS estiver registrado para uma coluna espacial específica, as coordenadas que são aplicáveis a ela poderão ser armazenadas na coluna associada da tabela CLIENTES. • Elimine o SRS denominado SRSDEMO1 Esta etapa será executada se você não precisar mais do SRS no banco de dados. Ao eliminar um SRS, você remove a definição do SRS do banco de dados. • Crie o SRS denominado KY_STATE_SRS
Criar e ocupar as tabelas espaciais	• Crie a tabela CLIENTES • Preencha a tabela CLIENTES A tabela CLIENTES representa dados de negócios que foram armazenados no banco de dados por vários anos. • Altere a tabela CLIENTES incluindo a coluna LOCALIZAÇÃO A instrução ALTER TABLE inclui uma nova coluna (LOCALIZAÇÃO) de tipo ST_Point. Esta coluna será ocupada, efetuando codificação geográfica das colunas de endereço em uma etapa subsequente. • Crie a tabela ESCRITÓRIOS A tabela ESCRITÓRIOS representa, entre outros dados, a zona de vendas para cada escritório de uma empresa de seguros. Toda a tabela será ocupada pelos dados do atributo a partir de um banco de dados não DB2 em uma etapa subsequente. Esta etapa subsequente envolve a importação de dados de atributo para a tabela ESCRITÓRIOS a partir de um arquivo shape.

Tabela 6. Etapas do Programa de Amostra do DB2 Spatial Extender (continuação)

Etapas do SQL	Ação e Descrição
Ocupar as colunas	• Codifique geograficamente os dados de endereço da coluna LOCALIZAÇÃO da tabela CLIENTES com o geocodificador denominado KY_STATE_GC Esta etapa executa codificação geográfica espacial em lote chamando o utilitário geocodificador. A codificação geográfica em lote geralmente é executada quando é necessário ser geocodificado ou geocodificado novamente de uma parte significativa da tabela. • Carregue a tabela ESCRITÓRIOS criada anteriormente a partir do arquivo shape, utilizando o sistema de referência espacial KY_STATE_SRS Esta etapa carrega a tabela ESCRITÓRIOS com dados espaciais existentes, na forma de um arquivo shape. Como a tabela ESCRITÓRIOS existe, o utilitário LOAD anexará os novos registros em uma tabela existente. • Crie e carregue a tabela ZONAS DE ALAGAMENTO a partir do arquivo shape, utilizando o sistema de referência espacial KY_STATE_SRS Esta etapa carrega a tabela ZONAS DE ALAGAMENTO com dados existentes, na formato de um arquivo shape. Como a tabela não existe, o utilitário LOAD a criará antes do carregamento dos dados. • Crie e carregue a tabela REGIÕES a partir do arquivo shape, utilizando o sistema de referência espacial KY_STATE_SRS
Registrar ou cancelar registro do geocodificador	• Registre o geocodificador SAMPLEGC • Cancele o registro do codificador denominado SAMPLEGC • Registre o codificador KY_STATE_GC Estas etapas registram e cancelam o registro do geocodificador denominado SAMPLEGC e, em seguida, criam um novo, KY_STATE_GC, para utilização no programa de amostra.
Criar índices espaciais	• Crie o índice de grade espacial para a coluna LOCALIZAÇÃO da tabela CLIENTES • Elimine o índice de grade espacial para a coluna LOCALIZAÇÃO da tabela CLIENTES • Crie o índice de grade espacial para a coluna LOCALIZAÇÃO da tabela CLIENTES • Crie o índice de grade espacial para a coluna LOCALIZAÇÃO da tabela ESCRITÓRIOS • Crie o índice de grade espacial para a coluna LOCALIZAÇÃO da tabela ZONAS DE ALAGAMENTO • Crie o índice de grade espacial para a coluna LOCALIZAÇÃO da tabela REGIÕES Estas etapas criam o índice de grade espacial para as tabelas CLIENTES, ESCRITÓRIOS, ZONAS DE ALAGAMENTO e REGIÕES.

Tabela 6. Etapas do Programa de Amostra do DB2 Spatial Extender (continuação)

Etapas do SQL	Ação e Descrição
Ativar codificação geográfica automática	• Configure o geocodificador para a coluna LOCALIZAÇÃO da tabela CLIENTES com o codificador KY_STATE_GC Esta etapa associa a coluna LOCALIZAÇÃO da tabela CLIENTES com o geocodificador KY_STATE_GC e configura os valores correspondentes para os parâmetros de codificação geográfica. • Ative a codificação geográfica automática para a coluna LOCALIZAÇÃO da tabela CLIENTES Esta etapa ativa a chamada automática do geocodificador. O uso de codificação geográfica automática faz as colunas LOCATION, LATITUDE e LONGITUDE da tabela CUSTOMERS serem sincronizadas entre si para operações subsequentes de inserção e atualização.
Execute operações de inserir, atualizar e excluir na tabela CLIENTES	Estas etapas demonstram operações de inserção, atualização e exclusão nas colunas LATITUDE, LONGITUDE, STREET, CITY, STATE e ZIP da tabela CUSTOMERS. Após a ativação da codificação geográfica automática, os dados inseridos ou atualizados nessas colunas são codificados automaticamente na coluna LOCALIZAÇÃO. Este processo foi ativado na etapa anterior. • Insira alguns registros com uma rua diferente • Atualize alguns registros com um novo endereço • Exclua todos os registros da tabela
Desativar codificação geográfica automática	Estas etapas desativam a chamada automática do geocodificador e o índice espacial na preparação para a próxima etapa. A próxima etapa envolve nova codificação geográfica de toda a tabela CLIENTES. • Desative a codificação geográfica automática para a coluna LOCALIZAÇÃO da tabela CLIENTES • Remova a configuração de codificação geográfica para a coluna LOCALIZAÇÃO da tabela CLIENTES • Elimine o índice espacial para a coluna LOCALIZAÇÃO da tabela CLIENTES **Recomendação:** Se estiver carregando uma grande quantidade de geodados, elimine o índice espacial antes de carregar os dados e, em seguida, recrie-o após o carregamento dos dados.
Criar uma exibição e registrar a coluna espacial na exibição	Estas etapas criam uma exibição e registram sua coluna espacial. • Crie uma exibição denominada CLIENTES DE ALTO RISCO, com base na junção das tabelas CLIENTES e ZONAS DE ALAGAMENTO • Registre a coluna espacial da exibição

Tabela 6. Etapas do Programa de Amostra do DB2 Spatial Extender (continuação)

Etapas do SQL	Ação e Descrição
Executar análise espacial	Estas etapas executam análise espacial utilizando os predicados espaciais e funções no DB2 SQL. O otimizador de consultas do DB2 explora o índice espacial nas colunas espaciais para aprimorar o desempenho de consultas sempre que possível. • Localize o número de clientes atendidos por cada região (ST_Within) • Para escritórios e clientes com a mesma região, localize o número de clientes que estão em uma distância específica de cada escritório (ST_Within, ST_Distance) • Para cada região, localize a renda média e adicional de cada cliente (ST_Within) • Localize o número de zonas de alagamento encontradas em cada zona de escritórios (ST_Overlaps) • Localize o escritório mais próximo de uma localização de cliente específica, assumindo que ele esteja localizado no centróide da zona de escritórios (ST_Distance) • Localize os clientes cuja localização esteja próxima do limite de uma zona de alagamento específica (ST_Buffer, ST_Intersects) • Localize os clientes de alto risco em um escritório especificado (ST_Within) Todas estas etapas utilizam a função interna gseRunSpatialQueries.
Exportar dados espaciais para arquivos shape	Esta etapa mostra um exemplo de exportação da exibição CLIENTES DE ALTO RISCO para arquivos shape. Exportar dados de um formato de banco de dados para outro formato de arquivo, permite que as informações sejam utilizadas por outras ferramentas (como o ArcExplorer para DB2). • Exporte a exibição CLIENTES DE ALTO RISCO para os arquivos shape

Capítulo 14. Identificando Problemas do DB2 Spatial Extender

Para identificar um problema do DB2 Spatial Extender, você deve determinar a causa do problema.

Você pode resolver os problemas com o DB2 Spatial Extender destas maneiras:

- Você pode utilizar informações de mensagens para diagnosticar o problema.
- Ao trabalhar com procedimentos armazenados e funções do Spatial Extender, o DB2 retorna informações sobre o êxito ou falha do procedimento armazenado ou função. As informações retornadas serão um código de mensagem (em forma de um inteiro), texto de mensagem, ou ambos, dependendo da interface utilizada para trabalhar com o DB2 Spatial Extender.
- Você pode exibir o arquivo de notificação de administração do DB2, que registra informações de diagnóstico sobre erros.
- Se tiver um problema recorrente e que pode ser reproduzido do Spatial Extender, um representante de suporte ao cliente IBM pode solicitar que você utilize o recurso de rastreio do DB2 para ajudá-lo a diagnosticar o problema.

Como Interpretar Mensagens do DB2 Spatial Extender

Entender como as mensagens do DB2 Spatial Extender são estruturadas e como obter informações adicionais para elas pode ajudá-lo a determinar se a operação espacial solicitada foi concluída com êxito ou resultou em um erro.

É possível trabalhar com o DB2 Spatial Extender usando qualquer uma das seguintes interfaces:

- Procedimentos armazenados do DB2 Spatial Extender
- Funções do DB2 Spatial Extender
- CLP (Processador da Linha de Comandos) do DB2 Spatial Extender

Todas estas interfaces retornam mensagens do DB2 Spatial Extender.

A tabela a seguir explica cada parte deste texto da mensagem do DB2 Spatial Extender de amostra:

```
GSE0000I: A operação foi concluída com êxito.
```

Tabela 7. As Partes do Texto da Mensagem do DB2 Spatial Extender

Parte do texto da mensagem	Descrição	
GSE	O identificador da mensagem. Todas as mensagens do DB2 Spatial Extender começam com o prefixo de três letras GSE.	
0000	O número da mensagem. Um número de quatro dígitos que varia de 0000 a 9999.	
I	O tipo de mensagem. Uma única letra que indica a gravidade da mensagem:	
	C	Mensagens de erro críticas
	N	Mensagens de erro não críticas
	W	Mensagens de aviso
	I	Mensagens informativas

Tabela 7. As Partes do Texto da Mensagem do DB2 Spatial Extender (continuação)

Parte do texto da mensagem	Descrição
A operação foi concluída com sucesso.	A explicação da mensagem.

A explicação que aparece no texto da mensagem é uma explicação breve. Você pode recuperar informações adicionais sobre a mensagem que incluem explicação detalhada e sugestões para evitar ou corrigir o problema. Para exibir estas informações adicionais:

1. Abra um prompt de comandos do sistema operacional.
2. Digite o comando de ajuda do DB2 com o identificador da mensagem e o número da mensagem para exibir informações adicionais sobre a mensagem. Por exemplo:

   ```
   DB2 "? GSEnnnn"
   ```

   em que *nnnn* é o número da mensagem.

Você pode digitar o identificador da mensagem GSE e a letra indicando o tipo de mensagem em maiúsculas ou minúsculas. Digitar `DB2 "? GSE0000I"` produzirá o mesmo resultado que digitar `db2 "? gse0000i"`.

Você pode omitir a letra após o número da mensagem quando digitar o comando. Por exemplo, digitar `DB2 "? GSE0000"` terá o mesmo resultado que digitar `DB2 "? GSE0000I"`.

Suponha que o código da mensagem seja `GSE4107N`. Ao digitar `DB2 "? GSE4107N"` no prompt de comandos, serão exibidas as seguintes informações:

```
GSE4107N O valor do tamanho da grade "<grid-size>" não é válido no local em que é
utilizado.

Explicação: O tamanho de grade especificado "<grid-size>" não é válido.

 Foi feita uma das seguintes especificações inválidas quando o índice de grade
 foi criado com a instrução CREATE INDEX:

  - Um número menor do que 0 (zero) foi especificado como o tamanho da
    grade para
    o primeiro, segundo ou terceiro nível de grade.

  - 0 (zero) foi especificado como o tamanho da grade para o primeiro
    nível de grade.

  - O tamanho da grade especificado para o segundo nível de grade é menor
    do que o
    tamanho da grade do primeiro nível de grade, mas não é 0 (zero).

  - O tamanho da grade especificado para o terceiro nível de grade é
    menor do que
    o tamanho da grade do segundo nível de grade, mas não
    é 0 (zero).

  - O tamanho da grade especificado para o terceiro nível de grade é maior
    do que 0 (zero)
    mas o tamanho da grade especificado para o segundo nível
    de grade é 0 (zero).

Resposta do Usuário: Especifique um valor válido para o tamanho da grade.
```

```
msgcode: -4107

sqlstate: 38SC7
```

Se as informações forem muito extensas para serem exibidas em uma única tela e seu sistema operacional suportar o programa executável **more** e canais, digite este comando:

```
db2 "? GSEnnnn" | more
```

A utilização do programa **more** forçará uma pausa na exibição após cada tela de dados para que você possa ler as informações.

Parâmetros de Saída de Procedimentos Armazenados do DB2 Spatial Extender

Use os parâmetros de saída de procedimento armazenado do DB2 Spatial Extender para diagnosticar problemas ao chamar explicitamente procedimentos armazenados em programas de aplicativo ou a partir do processador de linha de comandos do DB2.

Os procedimentos armazenados do DB2 Spatial Extender têm dois parâmetros de saída: o código da mensagem (msg_code) e o texto da mensagem (msg_text). Os valores de parâmetros indicam o êxito ou falha de um procedimento armazenado.

msg_code

O parâmetro msg_code é um inteiro, que pode ser positivo, negativo ou zero (0). Os números positivos são utilizados para avisos, os números negativos são utilizados para erros (críticos e não críticos) e zero (0) é utilizado para mensagens informativas.

O valor absoluto de msg_code está incluído em msg_text como o número da mensagem. Por exemplo

- Se msg_code for 0, o número da mensagem será 0000.
- Se msg_code for –219 , o número da mensagem será 0219. O msg_code negativo indica que uma mensagem é um erro crítico ou não crítico.
- Se msg_code for +1036, o número da mensagem será 1036. O número de msg_code positivo indica que a mensagem é um aviso.

Os números de msg_code para procedimentos armazenados do Spatial Extender estão divididos nas três categorias mostradas na tabela a seguir:

Tabela 8. Códigos de mensagens de procedimentos armazenados

Códigos	Categoria
0000 – 0999	Mensagens comuns
1000 - 1999	Mensagens administrativas
2000 - 2999	Mensagens de importação e exportação

msg_text

O parâmetro msg_text consiste no identificador de mensagem, no número da mensagem, no tipo de mensagem e na explicação. Um exemplo de um valor de msg_text de procedimento armazenado é:

```
GSE0219N   Uma instrução EXECUTE IMMEDIATE
           falhou. SQLERROR = "<sql-error>".
```

A explicação que aparece no parâmetro msg_text é a explicação breve. Você pode recuperar informações adicionais sobre a mensagem que incluem explicação detalhada e sugestões para evitar ou corrigir o problema.

Para obter uma explicação detalhada das partes do parâmetro msg_text, e informações sobre como recuperar informações adicionais sobre a mensagem, consulte o tópico: Como Interpretar Mensagens do DB2 Spatial Extender.

Para diagnosticar procedimentos armazenados chamados implicitamente emitindo comandos do DB2 Spatial Extender, use as mensagens retornadas pelo CLP do DB2 Spatial Extender. Para obter detalhes adicionais, consulte "Como Interpretar Mensagens do DB2 Spatial Extender" na página 103

Trabalhando com procedimentos armazenados em aplicativos

Quando chamar um procedimento armazenado do DB2 Spatial Extender a partir de um aplicativo, você receberá msg_code e msg_text como parâmetros de saída. Você pode:

- Programar seu aplicativo para retornar os valores de parâmetros de saída ao usuário do aplicativo.
- Executar alguma ação com base no tipo de valor de msg_code retornado.

Trabalhando com Procedimentos Armazenados a partir da Linha de Comandos do DB2

Quando chamar um procedimento armazenado do DB2 Spatial Extender a partir da linha de comandos do DB2, você receberá os parâmetros de saída msg_code e msg_text. Estes parâmetros de saída indicam o êxito ou falha do procedimento armazenado.

Suponha que você se conecte a um banco de dados e deseje chamar o procedimento ST_DISABLE_DB. O exemplo a seguir usa um comando DB2 CALL para desativar o banco de dados para operações espaciais e mostra os resultados do valor de saída. É utilizado um valor de parâmetro force 0, junto com dois pontos de interrogação no final do comando CALL para representar os parâmetros de saída msg_code e msg_text. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

```
call db2gse.st_disable_db(0, ?, ?)

  Valor de Parâmetros de Saída
  --------------------------
  Nome do Parâmetro  : MSGCODE
  Valor do Parâmetro : 0

  Nome do Parâmetro  : MSGTEXT
  Valor do Parâmetro : GSE0000I  A operação foi concluída com êxito.

  Status de Retorno = 0
```

Suponha que o msg_text retornado seja GSE2110N. Utilize o comando de ajuda do DB2 para exibir informações adicionais sobre a mensagem. Por exemplo:

```
"? GSE2110"
```

São exibidas as seguintes informações:

```
GSE2110N    O sistema de referência espacial para
            a geometria na linha "<row-number>" é invalida.
            O identificador numérico do sistema de referência espacial
```

```
            é "<srs-id>".

Explicação: Na linha número da linha, a geometria a
ser exportada utiliza um sistema de referência espacial inválido.
A geometria não pode ser exportada.

Resposta ao Usuário: Corrija a geometria indicada ou
exclua a linha da operação de exportação
modificando a instrução SELECT de acordo.

msg_code: -2110

sqlstate: 38S9A
```

Mensagens de Funções do DB2 Spatial Extender

As mensagens retornadas pelo DB2 Spatial Extender geralmente são integradas em uma mensagem SQL. O SQLCODE retornado na mensagem indica se ocorreu um erro com a função ou se um aviso está associado à função.

As mensagens a seguir são exemplos que indicam um erro e um aviso:

- O SQLCODE –443 (número de mensagem SQL0443) indica que ocorreu um erro na função.
- O SQLCODE +462 (número de mensagem SQL0462) indica que um aviso está associado à função.

A tabela a seguir explica as partes importantes desta mensagem de exemplo:

```
DB21034E  O comando foi processado como uma instrução SQL porque
ele não era um comando do Processador de Linha de Comandos válido.  Durante o
processamento SQL, ele
retornou: SQL0443N
 A rotina "DB2GSE.GSEGEOMFROMWKT"
(nome específico "GSEGEOMWKT1") retornou um erro
SQLSTATE com texto de diagnóstico "GSE3421N  O polígono não está fechado.".
SQLSTATE=38SSL
```

Tabela 9. As Partes Importantes de Mensagens de Funções do DB2 Spatial Extender

Parte da mensagem	Descrição
SQL0443N	O SQLCODE indica o tipo de problema.
GSE3421N	O número da mensagem e o tipo de mensagem do DB2 Spatial Extender. Os números de mensagens para funções vão de GSE3000 a GSE3999. Além disso, as mensagens comuns podem ser retornadas quando você trabalha com funções do DB2 Spatial Extender. Os números de mensagens para mensagens comuns vão de GSE0001 a GSE0999.
O polígono não está fechado	A explicação da mensagem do DB2 Spatial Extender.
SQLSTATE=38SSL	Um código SQLSTATE que identifica ainda mais o erro. É retornado um código SQLSTATE para cada instrução ou linha. • Os códigos SQLSTATE para erros de funções do Spatial Extender são 38S*xx*, em que cada *x* é uma letra ou um número. • Os códigos SQLSTATE para avisos de funções do Spatial Extender são 01HS*x*, em que o *x* é uma letra ou um número.

Um exemplo de uma mensagem de erro SQL0443

Suponha que você tente inserir os valores para um polígono na tabela POLYGON_TABLE, conforme mostrado na instrução a seguir:

```
INSERT INTO polygon_table ( geometry )
VALUES ( ST_Polygon ( 'polygon (( 0 0, 0 2, 2 2, 1 2)) ') )
```

Isto resulta em uma mensagem de erro porque você não forneceu o valor final para fechar o polígono. A mensagem de erro retornada é:

```
DB21034E  O comando foi processado como uma instrução SQL porque
ele não era um comando do Processador de Linha de Comandos válido.  Durante o
processamento SQL, ele
retornou: SQL0443N
 A rotina "DB2GSE.GSEGEOMFROMWKT"
(nome específico "GSEGEOMWKT1") retornou um erro
SQLSTATE com texto de diagnóstico "GSE3421N  O polígono não está fechado.".
SQLSTATE=38SSL
```

O número da mensagem SQL `SQL0443N` indica que ocorreu um erro e a mensagem inclui o texto da mensagem do Spatial Extender `GSE3421N O polígono não está fechado`.

Quando receber este tipo de mensagem:

1. Localize o número da mensagem GSE na mensagem de erro do DB2 ou SQL.
2. Utilize o comando da ajuda do DB2 (DB2 ?) para ver a explicação da mensagem do Spatial Extender e a resposta ao usuário. Usando este exemplo, digite o seguinte comando em um prompt da linha de comandos do sistema operacional:

   ```
   DB2 "? GSE3421"
   ```

A mensagem será repetida, junto com uma explicação detalhada e a resposta ao usuário recomendada.

Mensagens de CLP do DB2 Spatial Extender

As mensagens do CLP do DB2 Spatial Extender indicam o sucesso ou a falha de uma operação. Além disso, elas podem fornecer informações de formato.

O CLP do DB2 Spatial Extender retorna mensagens para:

- Procedimentos armazenados, se chamados implicitamente.
- Informações de formatos, se você chamou o programa de subcomando **shape_info** a partir do CLP do DB2 Spatial Extender. Estas são mensagens informativas.
- Operações de upgrade.
- Operações de importação e exportação de formatos para e a partir do cliente.

Exemplos de Mensagens para Procedimentos Armazenados Retornados pelo CLP do DB2 Spatial Extender

A maioria das mensagens retornadas pelo CLP do DB2 Spatial Extender destinam-se aos procedimentos armazenados do DB2 Spatial Extender. Quando chamar um procedimento armazenado a partir do CLP do DB2 Spatial Extender, você receberá um texto de mensagem que indica o êxito ou falha do procedimento armazenado.

O texto da mensagem consiste no identificador de mensagem, no número da mensagem, no tipo de mensagem e na explicação. Por exemplo, se você ativar um banco de dados utilizando o comando `db2se enable_db testdb`, o texto da mensagem retornada pelo CLP do Spatial Extender será:

```
Ativando o banco de dados. Aguarde...

GSE1036W Operação bem-sucedida.  Porém,
          alguns parâmetros de
          configuração do banco de dados e do gerenciador do
          parâmetros de configuração do banco de dados
          devem ser aumentados.
```

Da mesma forma, se você desativar um banco de dados utilizando o comando `db2se disable_db testdb`, o texto da mensagem retornada pelo CLP do Spatial Extender será:

```
GSE0000I  A operação foi concluída com êxito.
```

A explicação que aparece no texto da mensagem é uma explicação breve. Você pode recuperar informações adicionais sobre a mensagem que incluem explicação detalhada e sugestões para evitar ou corrigir o problema. As etapas para recuperar estas informações e uma explicação detalhada de como interpretar as partes do texto da mensagem são discutidas em um tópico separado.

Se estiver chamando procedimentos armazenados por meio de um programa aplicativo ou da linha de comandos do DB2, há um tópico separado que discute como diagnosticar os parâmetros de saída.

Exemplo de Mensagens para Informações de Formatos Retornadas pelo CLP do Spatial Extender

Suponha que você tenha decidido exibir informações para um arquivo modelo denominado `office`. Utilizando o CLP do Spatial Extender (db2se), você pode emitir este comando:

```
db2se shape_info -fileName /tmp/offices
```

Este é um exemplo das informações que são exibidas:

```
Informações do arquivo modelo
----------------------
Código do arquivo                           = 9994
Comprimento do arquivo (palavras de 16 bits) = 484
Versão do arquivo modelo                    = 1000
Tipo de formato                             = 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
Formatos não têm coordenadas Z.
Formatos não têm coordenadas M.

O arquivo shape de índice (extensão .shx) está presente.

Informações do arquivo de atributos
-------------------------
Código do arquivo dBase          = 3
Data da última atualização       = 1901-08-15
Número de registros              = 31
Número de bytes no cabeçalho     = 129
Número de bytes em cada registro = 39
```

```
Número de colunas                = 3

Número Coluna  Nome Coluna      Tipo de Dados     Compr.   Decimal

           1   NOME             C ( Caractere)      16        0
           2   FUNCIONÁRIOS     N (   Numérico™)    11        0
           3   ID               N (   Numérico)     11        0

Definição do 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]]"
```

Exemplos de Mensagens para Operações de Upgrade Retornadas pelo CLP do Spatial Extender

Quando você chama comandos que executam operações de upgrade, são retornadas mensagens que indicam o sucesso ou a falha dessa operação.

Suponha que você faça upgrade do banco de dados espacial ativado *mydb* usando o seguinte comando:

```
db2se upgrade mydb -messagesFile /tmp/db2se_upgrade.msg
```

O texto da mensagem retornada pelo CLP do Spatial Extender será:

```
Fazendo upgrade de banco de dados. Aguarde...
GSE0000I  A operação foi concluída com êxito.
```

Rastreio de Problemas no DB2 Spatial Extender com o Comando db2trc

Quando ocorrer um problema recorrente e que pode ser reproduzido no DB2 Spatial Extender, você pode utilizar o recurso de rastreio do DB2 para capturar informações sobre o problema.

Antes de Iniciar

Certifique-se de ter a autorização adequada para emitir o comando **db2trc**. Nos sistemas operacionais UNIX, você deve ter autorização SYSADM, SYSCTRL ou SYSMAINT para rastrear uma instância do DB2. Nos sistemas operacionais Windows, nenhuma autorização especial é necessária.

Sobre Esta Tarefa

O recurso de rastreio do DB2 é ativado pelo comando **db2trc**. O recurso de rastreio do DB2 pode:

- Rastrear eventos
- Efetuar dump dos dados de rastreio em um arquivo
- Formatar dados de rastreio em um formato legível

Restrições

- Ative este recurso somente quando indicado por um representante de suporte técnico do DB2.
- O comando **db2trc** deve ser inserido em um prompt de comandos do sistema operacional ou em um shell script. Ele não pode ser usado na interface da linha de comandos do DB2 Spatial Extender (**db2se**) ou no CLP do DB2.

Procedimento

Para solucionar problemas no DB2 Spatial Extender usando o recurso de rastreio do DB2:

1. Encerre todos os outros aplicativos.
2. Ative o rastreio.

   O representante de suporte técnico do Suporte DB2 fornece os parâmetros específicos para esta etapa. O comando básico é:

   ```
   db2trc on
   ```

   Você pode rastrear para a memória ou para um arquivo. O melhor método de rastreio é para a memória. Se o problema que está sendo recriado suspender a estação de trabalho e impedir o dump do rastreio, rastreie para um arquivo.
3. Reproduza o problema.
4. Efetue dump do rastreio para um arquivo imediatamente após o problema ocorrer.

   Por exemplo:

   ```
   db2trc dump january23trace.dmp
   ```

   Esse comando cria um arquivo (`january23trace.dmp`) no diretório atual, com o nome especificado, e efetua dump das informações de rastreio nesse arquivo. Você pode especificar um diretório diferente, incluindo o caminho do arquivo. Por exemplo, para colocar o arquivo de dump no diretório `/tmp/spatial/errors`, a sintaxe é:

   ```
   db2trc dump /tmp/spatial/errors/january23trace.dmp
   ```
5. Desative o rastreio.

   Por exemplo:

   ```
   db2trc off
   ```
6. Formate os dados como um arquivo ASCII. Você pode classificar os dados de duas formas:
   - Utilize a opção **flw** para classificar os dados por processo ou por encadeamento. Por exemplo:

     ```
     db2trc flw january23trace.dmp january23trace.flw
     ```
   - Utilize a opção **fmt** para relacionar os eventos em ordem cronológica. Por exemplo:

     ```
     db2trc fmt january23trace.dmp january23trace.fmt
     ```

O Arquivo de Notificação de Administração

As informações de diagnóstico sobre os erros são registradas no arquivo de notificação de administração. Essas informações são usadas para a determinação de problemas e destinam-se ao suporte técnico do DB2.

O arquivo de notificação de administração contém informações em texto registradas pelo sistema de banco de dados DB2 e também pelo DB2 Spatial Extender. Ele está localizado no diretório especificado pelo parâmetro de configuração do gerenciador de banco de dados **diagpath**. Nos sistemas operacionais Windows, o arquivo de notificação de administração do DB2 está localizado no log de eventos e pode ser revisado através do Windows Event Viewer.

As informações que o gerenciador de banco de dados DB2 registra no log de administração são determinadas pelas configurações **diaglevel** e **notifylevel**.

Utilize um editor de texto para exibir o arquivo na máquina na qual você suspeita que ocorreu um problema. Os eventos mais recentes registrados são os últimos do arquivo. Geralmente, cada entrada contém as seguintes partes:

- Uma data e hora.
- A localização que relata o erro. Os identificadores do aplicativo permitem corresponder entradas relativas a um aplicativo nos logs de servidores e clientes.
- Uma mensagem de diagnóstico (geralmente iniciando com "DIA" ou "ADM") explicando o erro.
- Dados de suporte disponíveis, como estruturas de dados SQLCA e ponteiros para a localização de arquivos dump ou trap extra.

Se o banco de dados estiver com comportamento normal, esse tipo de informação não é importante e pode ser ignorado.

O arquivo de notificação de administração cresce continuamente. Quando ficar muito grande, faça backup e apague-o. Um novo arquivo será gerado automaticamente na próxima vez que o sistema precisar dele.

Capítulo 15. Exibições do catálogo

É possível usar visualizações de catálogo para o Spatial Extender para obter informações úteis sobre dados espaciais.

As visualizações de catálogo do Spatial Extender contêm informações sobre:

"visualização de catálogo DB2GSE.ST_COORDINATE_ SYSTEMS"
Sistemas de coordenadas que você pode utilizar

"Exibição do catálogo DB2GSE.ST_GEOMETRY_COLUMNS" na página 114
Colunas espaciais que você pode preencher ou atualizar.

"exibição do catálogo DB2GSE.ST_GEOCODERS" na página 117 e "exibição do catálogo DB2GSE.ST_GEOCODER_ PARAMETERS" na página 115
Geocodificadores que você pode utilizar

"exibição do catálogo DB2GSE.ST_GEOCODING" na página 117 e "exibição do catálogo DB2GSE.ST_GEOCODING_ PARAMETERS" na página 118
Especificações para configurar um geocodificador para ser executado automaticamente e para definir, antecipadamente, operações a serem executadas durante a codificação geográfica em lote.

"visualização de catálogo DB2GSE.ST_SIZINGS" na página 120
Comprimentos máximos permitidos para os valores que você pode atribuir a variáveis.

"Exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS" na página 121
Sistemas de referência espacial que podem ser usados.

"visualização de catálogo DB2GSE.ST_UNITS_OF_ MEASURE" na página 123
As unidades de medida (metros, milhas, pés e assim por diante) nas quais as distâncias geradas por funções espaciais podem ser expressas.

visualização de catálogo DB2GSE.ST_COORDINATE_ SYSTEMS

Consulte a visualização de catálogo DB2GSE.ST_COORDINATE_SYSTEMS para recuperar informações sobre sistemas de coordenadas registrados.

O Spatial Extender registra automaticamente sistemas de coordenadas no catálogo do Spatial Extender nas seguintes situações:

- Quando ativar um banco de dados para operações espaciais.
- Quando os usuários definem sistemas de coordenadas adicionais para o banco de dados.

Para obter uma descrição de colunas nesta visualização, consulte a tabela a seguir.

Tabela 10. Colunas na visualização de catálogo DB2GSE.ST_COORDINATE_SYSTEMS

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
COORDSYS_NAME	VARCHAR(128)	Não	Nome deste sistema coordenado. O nome é exclusivo no banco de dados.

Tabela 10. Colunas na visualização de catálogo DB2GSE.ST_COORDINATE_SYSTEMS (continuação)

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
COORDSYS_TYPE	VARCHAR(128)	Não	Tipo deste sistema de coordenadas: **PROJECTED** Bidimensional. **GEOGRAPHIC** Tridimensional. Utiliza coordenadas X e Y. **GEOCENTRIC** Tridimensional. Utiliza coordenadas X, Y e Z. **UNSPECIFIED** Sistema de coordenadas abstrato ou irreal. O valor para esta coluna é obtido da coluna DEFINITION.
DEFINITION	VARCHAR(2048)	Não	Representação de texto convencional da definição deste sistema de coordenadas.
ORGANIZATION	VARCHAR(128)	Sim	Nome da organização (por exemplo, um órgão de padronizações como o European Petrol Survey Group, ou ESPG) que definiu este sistema de coordenadas. Esta coluna será nula se a coluna ORGANIZATION_COORDSYS_ID for nula.
ORGANIZATION_ COORDSYS_ID	INTEGER	Sim	Identificador numérico atribuído a este sistema de coordenadas pela organização que definiu o sistema de coordenadas. Este identificador e o valor na coluna ORGANIZATION identificam exclusivamente o sistema de coordenadas, a menos que o identificador e o valor sejam nulos. Se a coluna ORGANIZATION for nula, a coluna ORGANIZATION_COORDSYS_ID também será nula.
DESCRIPTION	VARCHAR(256)	Sim	Descrição do sistema de coordenadas que indica seu aplicativo.

Exibição do catálogo DB2GSE.ST_GEOMETRY_COLUMNS

Utilize a exibição do catálogo DB2GSE.ST_GEOMETRY_COLUMNS para encontrar informações sobre todas as colunas espaciais em todas as tabelas que contêm dados espaciais no banco de dados.

Se uma coluna espacial foi registrada junto com um sistema de referência espacial, será possível também usar a visualização para descobrir o nome e o identificador numérico do sistema de referência espacial. Para obter informações adicionais sobre colunas espaciais, consulte a visualização de catálogo SYSCAT.COLUMN.

Se uma coluna espacial foi registrada junto com um sistema de referência espacial e **compute_extents** foi especificado, também será possível usar esta visualização para consultar informações sobre as extensões geográficas para a coluna espacial e a data de seu último cálculo.

Para obter uma descrição do DB2GSE.ST_GEOMETRY_COLUMNS, consulte a tabela a seguir.

Tabela 11. Colunas na exibição do catálogo DB2GSE.ST_GEOMETRY_COLUMNS

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
TABLE_SCHEMA	VARCHAR(128)	Não	Nome do esquema ao qual pertence a tabela que contém esta coluna espacial.
TABLE_NAME	VARCHAR(128)	Não	Nome não qualificado da tabela que contém esta coluna espacial.
COLUMN_NAME	VARCHAR(128)	Não	Nome desta coluna espacial. A combinação de TABLE_SCHEMA, TABLE_NAME e COLUMN_NAME que identifica exclusivamente a coluna.
TYPE_SCHEMA	VARCHAR(128)	Não	Nome do esquema ao qual pertence o tipo de dados declarado desta coluna espacial. Este nome é obtido do catálogo do DB2.
TYPE_NAME	VARCHAR(128)	Não	Nome não qualificado do tipo de dados declarado desta coluna espacial. Este nome é obtido do catálogo do DB2.
SRS_NAME	VARCHAR(128)	Sim	Nome do sistema de referência espacial que está associado a esta coluna espacial. Se nenhum sistema de referência espacial estiver associado à coluna, SRS_NAME será nulo.
SRS_ID	INTEGER	Sim	Identificador numérico do sistema de referência espacial que está associado a esta coluna espacial. Se nenhum sistema de referência espacial estiver associado à coluna, SRS_ID será nulo.
MIN_X	DOUBLE	Sim	Valor *x* mínimo de todos os valores espaciais na coluna.
MIN_Y	DOUBLE	Sim	Valor *y* mínimo de todos os valores espaciais na coluna.
MIN_Z	DOUBLE	Sim	Valor *z* mínimo de todos os valores espaciais na coluna.
MAX_X	DOUBLE	Sim	Valor *x* máximo de todos os valores espaciais na coluna.
MAX_Y	DOUBLE	Sim	Valor *y* máximo de todos os valores espaciais na coluna.
MAX_Z	DOUBLE	Sim	Valor *z* máximo de todos os valores espaciais na coluna.
MIN_M	DOUBLE	Sim	Valor *m* mínimo de todos os valores espaciais na coluna.
MAX_M	DOUBLE	Sim	Valor *m* máximo de todos os valores espaciais na coluna.
EXTENT_TIME	TIMESTAMP	Sim	Registro de data e hora do último cálculo das extensões.

exibição do catálogo DB2GSE.ST_GEOCODER_ PARAMETERS

Use a visualização de catálogo DB2GSE.ST_GEOCODER_PARAMETERS para recuperar informações sobre os parâmetros de geocodificadores registrados.

Para obter informações adicionais sobre parâmetros de geocodificadores, consulte a visualização de catálogo SYSCAT.ROUTINEPARMS.

Para obter uma descrição de colunas nesta visualização, consulte a tabela a seguir.

Tabela 12. Colunas em DB2GSE.ST_GEOCODER_PARAMETERS

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
GEOCODER_NAME	VARCHAR(128)	Não	Nome do geocodificador ao qual este parâmetro pertence.
ORDINAL	SMALLINT	Não	A posição deste parâmetro (isto é, o parâmetro especificado na coluna PARAMETER_NAME) na assinatura da função que serve como o geocodificador especificado na coluna GEOCODER_NAME. Os valores combinados nas colunas GEOCODER_NAME e ORDINAL identificam exclusivamente este parâmetro. Um registro na visualização de catálogo SYSCAT.ROUTINEPARMS também contém informações sobre este parâmetro. Este registro contém um valor que aparece na coluna ORDINAL de SYSCAT.ROUTINEPARMS. Este valor é igual ao que aparece na coluna ORDINAL da exibição DB2GSE.ST_GEOCODER_PARAMETERS.
PARAMETER_NAME	VARCHAR(128)	Sim	Nome deste parâmetro. Se não foi especificado um nome durante a criação da função à qual este parâmetro pertence, a coluna PARAMETER_NAME será nula. O conteúdo da coluna PARAMETER_NAME é obtido no catálogo do DB2.
TYPE_SCHEMA	VARCHAR(128)	Não	Nome do esquema ao qual este parâmetro pertence. Este nome é obtido do catálogo do DB2.
TYPE_NAME	VARCHAR(128)	Não	Nome não qualificado do tipo de dados dos valores atribuídos a este parâmetro. Este nome é obtido do catálogo do DB2.
PARAMETER_DEFAULT	VARCHAR(2048)	Sim	O valor padrão que deve ser atribuído a este parâmetro. O DB2 interpretará este valor como uma expressão SQL. Se o valor estiver entre aspas, ele será transmitido para o geocodificador como uma sequência. Caso contrário, a avaliação da expressão SQL determinará qual será o tipo de dados do parâmetro quando ele for transmitido para o geocodificador. Se a coluna PARAMETER_DEFAULT contiver um nulo, este valor nulo será transmitido para o geocodificador. O valor padrão deve ter um valor correspondente na exibição do catálogo DB2GSE.ST_GEOCODING_PARAMETERS. Ele também pode ter um valor correspondente nos parâmetros de entrada para o procedimento ST_RUN_GEOCODING. Se qualquer um dos valores correspondentes for diferente do valor padrão, o valor correspondente substituirá o valor padrão.
DESCRIPTION	VARCHAR(256)	Sim	Descrição do parâmetro que indica seu aplicativo.

exibição do catálogo DB2GSE.ST_GEOCODERS

Quando desejar disponibilizar geocodificadores adicionais para usuários, você precisa registrar estes geocodificadores. Para recuperar informações sobre geocodificadores registrados, consulte a exibição do catálogo DB2GSE.ST_GEOCODERS.

Para obter informações sobre parâmetros de geocodificadores, consulte a visualização de catálogo DB2GSE.ST_GEOCODER_PARAMETERS e a visualização de catálogo SYSCAT.ROUTINEPARMS. Para obter informações sobre as funções que são usadas como geocodificadores, consulte a visualização de catálogo SYSCAT.ROUTINES.

Para obter uma descrição de colunas na visualização DB2GSE.ST_GEOCODERS, consulte a tabela a seguir.

Tabela 13. Colunas na exibição do catálogo DB2GSE.ST_GEOCODERS

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
GEOCODER_NAME	VARCHAR(128)	Não	Nome deste geocodificador. Ele é exclusivo no banco de dados.
FUNCTION_SCHEMA	VARCHAR(128)	Não	Nome do esquema ao qual pertence a função que está sendo utilizada como este geocodificador.
FUNCTION_NAME	VARCHAR(128)	Não	Nome não qualificado da função que está sendo utilizada como este geocodificador.
SPECIFIC_NAME	VARCHAR(128)	Não	Nome específico da função que está sendo utilizada como este geocodificador. Os valores combinados de FUNCTION_SCHEMA e SPECIFIC_NAME identificam exclusivamente a função que está sendo utilizada como este geocodificador.
RETURN_TYPE_SCHEMA	VARCHAR(128)	Não	Nome do esquema ao qual pertence o tipo de dados deste parâmetro de saída do geocodificador. Este nome é obtido do catálogo do DB2.
RETURN_TYPE_NAME	VARCHAR(128)	Não	Nome não qualificado do tipo de dados deste parâmetro de saída do geocodificador. Este nome é obtido do catálogo do DB2.
VENDOR	VARCHAR(256)	Sim	Nome do fornecedor que criou este geocodificador.
DESCRIPTION	VARCHAR(256)	Sim	Descrição do geocodificador que indica seu aplicativo.

exibição do catálogo DB2GSE.ST_GEOCODING

Ao configurar operações de geocodificação, as características particulares de suas definições serão automaticamente registradas no catálogo do DB2 Spatial Extender. Para saber quais são estas características particulares, consulte as exibições do catálogo DB2GSE.ST_GEOCODING e DB2GSE.ST_GEOCODING_PARAMETERS.

A visualização de catálogo DB2GSE.ST_GEOCODING, que é descrita na Tabela 14 na página 118, contém detalhes de todas as configurações; por exemplo, o número de registros que um geocodificador deve processar antes de cada confirmação.

A exibição do catálogo DB2GSE.ST_GEOCODING_PARAMETERS contém características particulares que são específicas de cada geocodificador. Por exemplo, o grau mínimo ao qual os endereços especificados como entrada e os endereços

reais devem corresponder para que o geocodificador geocodifique a entrada. Este requisito mínimo, chamado de score mínimo de correspondência, é registrado na exibição do catálogo DB2GSE.ST_GEOCODING_PARAMETERS.

Tabela 14. Colunas na exibição do catálogo DB2GSE.ST_GEOCODING

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
TABLE_SCHEMA	VARCHAR(128)	Não	Nome do esquema que contém a tabela que contém a coluna identificada na coluna COLUMN_NAME.
TABLE_NAME	VARCHAR(128)	Não	Nome não qualificado da tabela que contém a coluna identificada na coluna COLUMN_NAME.
COLUMN_NAME	VARCHAR(128)	Não	Nome da coluna espacial a ser ocupada de acordo com as especificações mostradas nesta exibição do catálogo. Os valores combinados nas colunas TABLE_SCHEMA, TABLE_NAME e COLUMN_NAME identificam exclusivamente a coluna espacial.
GEOCODER_NAME	VARCHAR(128)	Não	Nome do geocodificador que deve gerar dados para a coluna espacial especificada na coluna COLUMN_NAME. Somente um geocodificador pode ser atribuído a uma coluna espacial.
MODE	VARCHAR(128)	Não	Modo para o processo de codificação geográfica: **BATCH** Somente o geocoding em lote está ativado. **AUTO** A codificação geográfica automática está configurada e ativada. **INVALID** Foi detectada uma inconsistência nas tabelas do catálogo espacial; a entrada de codificação geográfica é inválida.
SOURCE_COLUMNS	VARCHAR(10000)	Sim	Nomes de colunas da tabela configuradas para codificação geográfica automática. Sempre que estas colunas forem atualizadas, um disparo solicitará ao geocodificador que efetue geocode dos dados atualizados.
WHERE_CLAUSE	VARCHAR(10000)	Sim	Condição de pesquisa em uma cláusula WHERE. Esta condição indica que quando o geocodificador é executado no modo em lote, ele efetua geocode somente dos dos em um subconjunto de registros especificado.
COMMIT_COUNT	INTEGER	Sim	O número de linhas que devem ser processadas durante a codificação geográfica em lote antes da emissão de uma consolidação. Se o valor na coluna COMMIT_COUNT for 0 (zero) ou nulo, nenhuma consolidação será emitida.

exibição do catálogo DB2GSE.ST_GEOCODING_ PARAMETERS

Quando você configura operações de codificação geográfica para um determinado geocodificador, os aspectos específicos do geocodificador das configurações ficam disponíveis por meio da visualização de catálogo DB2GSE.ST_GEOCODING_PARAMETERS.

Por exemplo, uma operação pode ser a comparação de endereços especificados como entrada para dados de referência.

Ao configurar operações para um geocodificador, você especifica qual deve ser o grau, chamado de pontuação de correspondência mínima, e sua especificação é registrada no catálogo.

Para saber os aspectos específicos do geocodificador das definições para operações de codificação geográfica, consulte a exibição do catálogo DB2GSE.ST_GEOCODING_PARAMETERS. Esta exibição é descrita na tabela a seguir.

Alguns padrões para configurações de operações de codificação geográfica estão disponíveis na exibição do catálogo DB2GSE.ST_GEOCODER_PARAMETERS. Os valores na exibição DB2GSE.ST_GEOCODING_PARAMETERS substituem os padrões.

Tabela 15. Colunas na exibição do catálogo DB2GSE.ST_GEOCODING_PARAMETERS

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
TABLE_SCHEMA	VARCHAR(128)	Não	Nome do esquema que contém a tabela que contém a coluna identificada na coluna COLUMN_NAME.
TABLE_NAME	VARCHAR(128)	Não	Nome não qualificado da tabela que contém a coluna espacial.
COLUMN_NAME	VARCHAR(128)	Não	Nome da coluna espacial a ser ocupada de acordo com as especificações mostradas nesta exibição do catálogo. Os valores combinados nas colunas TABLE_SCHEMA, TABLE_NAME e COLUMN_NAME identificam exclusivamente esta coluna espacial.
ORDINAL	SMALLINT	Não	A posição deste parâmetro (ou seja, o parâmetro especificado na coluna PARAMETER_NAME) na assinatura da função que serve como o geocodificador para a coluna identificada na coluna COLUMN_NAME. Um registro na visualização de catálogo SYSCAT.ROUTINEPARMS também contém informações sobre este parâmetro. Este registro contém um valor que aparece na coluna ORDINAL de SYSCAT.ROUTINEPARMS. Este valor é igual ao que aparece na coluna ORDINAL da exibição DB2GSE.ST_GEOCODING_PARAMETERS.
PARAMETER_NAME	VARCHAR(128)	Sim	Nome de um parâmetro na definição do geocodificador. Se nenhum nome foi especificado quando o geocodificador foi definido, PARAMETER_NAME será nulo. Este conteúdo da coluna PARAMETER_NAME é obtido no catálogo do DB2.

Tabela 15. Colunas na exibição do catálogo DB2GSE.ST_GEOCODING_PARAMETERS (continuação)

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
PARAMETER_VALUE	VARCHAR(2048)	Sim	O valor que é atribuído a este parâmetro. O DB2 interpretará este valor como uma expressão SQL. Se o valor estiver entre aspas, ele será transmitido para o geocodificador como uma sequência. Caso contrário, a avaliação da expressão SQL determinará qual será o tipo de dados do parâmetro quando ele for transmitido para o geocodificador. Se a coluna PARAMETER_VALUE contiver um nulo, este nulo será transmitido para o geocodificador. A coluna PARAMETER_VALUE corresponde à coluna PARAMETER_DEFAULT na exibição do catálogo DB2GSE.ST_GEOCODER_PARAMETERS. Se a coluna PARAMETER_VALUE contiver um valor, este valor substituirá o valor padrão na coluna PARAMETER_DEFAULT. Se a coluna PARAMETER_VALUE for nula, será utilizado o valor padrão.

visualização de catálogo DB2GSE.ST_SIZINGS

Use a visualização de catálogo DB2GSE.ST_SIZINGS para recuperar informações sobre variáveis suportadas e seu comprimento máximo.

Esta visualização de catálogo retorna as seguintes informações:

- Todas as variáveis suportadas pelo Spatial Extender; por exemplo, nome do sistema de coordenadas, nome do geocodificador e variáveis às quais as representações de texto convencional de dados espaciais podem ser atribuídas.
- O comprimento máximo permitido, se conhecido, de valores atribuídos a estas variáveis (por exemplo, os comprimentos máximos permitidos de nomes de sistemas de coordenadas, de nomes de geocodificadores e de representações de texto convencional de dados espaciais).

Para obter uma descrição de colunas nesta visualização, consulte a tabela a seguir.

Tabela 16. Colunas na visualização de catálogo DB2GSE.ST_SIZINGS

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
VARIABLE_NAME	VARCHAR(128)	Não	Termo que indica uma variável. O termo é exclusivo no banco de dados.

Tabela 16. Colunas na visualização de catálogo DB2GSE.ST_SIZINGS (continuação)

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
SUPPORTED_VALUE	INTEGER	Sim	Comprimento máximo permitido dos valores atribuídos à variável mostrada na coluna VARIABLE_NAME. Os possíveis valores na coluna SUPPORTED_VALUE são: **Um valor numérico diferente de 0** O comprimento máximo permitido de valores atribuídos a esta variável. **0** Qualquer comprimento é permitido, ou o comprimento permitido não pode ser determinado. **NULL** O Spatial Extender não suporta esta variável.
DESCRIPTION	VARCHAR(128)	Sim	Descrição desta variável.

Exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS

Consulte a exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS para recuperar informações sobre sistemas de referência espacial registrados.

O Spatial Extender registra automaticamente sistemas de referência espacial no catálogo do Spatial Extender nas seguintes situações:

- Ao ativar um banco de dados para operações espaciais, ele registra cinco sistemas de referência espacial padrão.
- Quando os usuários criam sistemas de referência espacial adicionais.

Para obter o valor completo da exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, é necessário entender que cada sistema de referência espacial está associado a um sistema de coordenadas. O sistema de referência espacial é projetado parcialmente para converter coordenadas derivadas do sistema de coordenadas em valores que o DB2 pode processar com o máximo de eficiência e, parcialmente, para definir a máxima extensão possível de espaço que pode ser referido por essas coordenadas.

Para saber o nome e tipo do sistema de coordenadas associado a um determinado sistema de referência espacial, consulte as colunas COORDSYS_NAME e COORDSYS_TYPE da exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS. Para obter mais informações sobre o sistema de coordenadas, consulte a exibição do catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Tabela 17. Colunas na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
SRS_NAME	VARCHAR(128)	Não	Nome do sistema de referência espacial. Este nome é exclusivo no banco de dados.

Tabela 17. Colunas na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS (continuação)

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
SRS_ID	INTEGER	Não	Identificador numérico do sistema de referência espacial. Cada sistema de referência espacial possui um identificador numérico exclusivo. As funções espaciais especificam sistemas de referência espacial por seus identificadores numéricos em vez de especificar por seus nomes.
X_OFFSET	DOUBLE	Não	O deslocamento a ser subtraído de todas as coordenadas X de uma geometria. A subtração é uma etapa no processo de conversão de coordenadas de geometria em valores que o DB2 pode processar com eficiência máxima. Uma etapa seguinte é multiplicar a figura resultante da subtração pelo fator de escala mostrado na coluna X_SCALE.
X_SCALE	DOUBLE	Não	Fator de escala pelo qual é multiplicada a figura resultante quando um deslocamento é subtraído de uma coordenada X. Este fator é idêntico ao valor mostrado na coluna Y_SCALE.
Y_OFFSET	DOUBLE	Não	O deslocamento a ser subtraído de todas as coordenadas Y de uma geometria. A subtração é uma etapa no processo de conversão de coordenadas de geometria em valores que o DB2 pode processar com eficiência máxima. Uma etapa seguinte é multiplicar a figura resultante da subtração pelo fator de escala mostrado na coluna Y_SCALE.
Y_SCALE	DOUBLE	Não	Fator de escala pelo qual é multiplicada a figura resultante quando um deslocamento é subtraído de uma coordenada Y. Este fator é idêntico ao valor mostrado na coluna X_SCALE.
Z_OFFSET	DOUBLE	Não	O deslocamento a ser subtraído de todas as coordenadas Z de uma geometria. A subtração é uma etapa no processo de conversão de coordenadas de geometria em valores que o DB2 pode processar com eficiência máxima. Uma etapa seguinte é multiplicar a figura resultante da subtração pelo fator de escala mostrado na coluna Z_SCALE.
Z_SCALE	DOUBLE	Não	Fator de escala pelo qual é multiplicada a figura resultante quando um deslocamento é subtraído de uma coordenada Z.
M_OFFSET	DOUBLE	Não	O deslocamento a ser subtraído de todas as medidas associadas a uma geometria. A subtração é uma etapa no processo de conversão das medidas em valores que o DB2 pode processar com eficiência máxima. Uma etapa seguinte é multiplicar a figura resultante da subtração pelo fator de escala mostrado na coluna M_SCALE.
M_SCALE	DOUBLE	Não	Fator de escala pelo qual é multiplicada a figura resultante quando um deslocamento é subtraído de uma medida.
MIN_X	DOUBLE	Não	Valor mínimo possível para coordenadas X nas geometrias às quais este sistema de referência espacial se aplica. Este valor é derivado dos valores nas colunas X_OFFSET e X_SCALE.

Tabela 17. Colunas na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS (continuação)

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
MAX_X	DOUBLE	Não	Valor máximo possível para coordenadas X nas geometrias às quais este sistema de referência espacial se aplica. Este valor é derivado dos valores nas colunas X_OFFSET e X_SCALE.
MIN_Y	DOUBLE	Não	Valor mínimo possível para coordenadas Y nas geometrias às quais este sistema de referência espacial se aplica. Este valor é derivado dos valores nas colunas Y_OFFSET e Y_SCALE.
MAX_Y	DOUBLE	Não	Valor máximo possível para coordenadas Y nas geometrias às quais este sistema de referência espacial se aplica. Este valor é derivado dos valores nas colunas Y_OFFSET e Y_SCALE.
MIN_Z	DOUBLE	Não	Valor mínimo possível para coordenadas Z nas geometrias às quais este sistema de referência espacial se aplica. Este valor é derivado dos valores nas colunas Z_OFFSET e Z_SCALE.
MAX_Z	DOUBLE	Não	Valor máximo possível para coordenadas Z nas geometrias às quais este sistema de referência espacial se aplica. Este valor é derivado dos valores nas colunas Z_OFFSET e Z_SCALE.
MIN_M	DOUBLE	Não	Valor mínimo possível para as medidas que podem ser armazenadas com as geometrias às quais este sistema de referência espacial se aplica. Este valor é derivado dos valores nas colunas M_OFFSET e M_SCALE.
MAX_M	DOUBLE	Não	Valor máximo possível para as medidas que podem ser armazenadas com as geometrias às quais este sistema de referência espacial se aplica. Este valor é derivado dos valores nas colunas M_OFFSET e M_SCALE.
COORDSYS_NAME	VARCHAR(128)	Não	Nome de identificação do sistema de coordenadas no qual este sistema de referência espacial está baseado.
COORDSYS_TYPE	VARCHAR(128)	Não	Tipo do sistema de coordenadas no qual este sistema de referência espacial está baseado.
ORGANIZATION	VARCHAR(128)	Sim	Nome da organização (por exemplo, um órgão de padronizações) que definiu o sistema de coordenadas no qual este sistema de referência espacial está baseado. ORGANIZATION será nulo se ORGANIZATION_COORSYS_ID for nulo.
ORGANIZATION_ COORDSYS_ID	INTEGER	Sim	Nome da organização (por exemplo, um órgão de padronizações) que definiu o sistema de coordenadas no qual este sistema de referência espacial está baseado. ORGANIZATION_COORDSYS_ID será nulo se ORGANIZATION for nulo.
DEFINITION	VARCHAR(2048)	Não	Representação de texto convencional da definição do sistema de coordenadas.
DESCRIPTION	VARCHAR(256)	Sim	Descrição do sistema de referência espacial.

visualização de catálogo DB2GSE.ST_UNITS_OF_ MEASURE

Consulte a visualização de catálogo DB2GSE.ST_UNITS_OF_MEASURE para determinar entre quais unidades de medida você escolherá.

Determinadas funções espaciais aceitam ou retornam valores que indicam uma distância específica. Em alguns casos, você pode escolher em qual unidade de medida a distância deve ser expressada. Por exemplo, ST_Distance retorna a distância mínima entre duas geometrias especificadas. Em uma ocasião, você pode requerer que ST_Distance retorne a distância em milhas; em outra, pode requerer uma distância expressa em metros.

Para obter uma descrição de colunas nesta visualização, consulte a tabela a seguir.

Tabela 18. Colunas na visualização de catálogo DB2GSE.ST_UNITS_OF_MEASURE

Nome	Tipo de dados	Pode ser anulada?	Conteúdo
UNIT_NAME	VARCHAR(128)	Não	Nome da unidade de medida. Este nome é exclusivo no banco de dados.
UNIT_TYPE	VARCHAR(128)	Não	Tipo da unidade de medida. Os possíveis valores são: **LINEAR** A unidade de medida é linear. **ANGULAR** A unidade de medida é angular.
CONVERSION_FACTOR	DOUBLE	Não	Valor numérico utilizado para converter esta unidade de medida em sua unidade base. A unidade base para unidades lineares de medida é METER; a unidade base para unidades angulares de medida é RADIAN. A própria unidade base tem um fator de conversão de 1.0.
DESCRIPTION	VARCHAR(256)	Sim	Descrição da unidade de medida.

Visualização de Catálogo DB2GSE.SPATIAL_REF_SYS

Use a visualização de catálogo DB2GSE.SPATIAL_REF_SYS para obter informações sobre sistemas de referência espacial que estão registrados no DB2 Spatial Extender.

Importante: Esta visualização de catálogo foi descontinuada e substituída pela "Exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS" na página 121.

Para obter uma descrição de colunas nesta visualização, consulte a tabela a seguir.

Tabela 19. Colunas na Visualização de Catálogo DB2GSE.SPATIAL_REF_SYS

Nome	Tipo de Dados	Pode ser anulada?	Conteúdo
SRID	INTEGER	Não	Identificador definido pelo usuário para este sistema de referência espacial.
SR_NAME	VARCHAR(64)	Não	Nome deste sistema de referência espacial.
CSID	INTEGER	Não	Identificador numérico para o sistema coordenado que suporta este sistema de referência espacial.
CS_NAME	VARCHAR(64)	Não	Nome do sistema coordenado que suporta este sistema de referência espacial.
AUTH_NAME	VARCHAR(256)	Sim	Nome da organização que define os padrões para este sistema de referência espacial.

Tabela 19. Colunas na Visualização de Catálogo DB2GSE.SPATIAL_REF_SYS (continuação)

Nome	Tipo de Dados	Pode ser anulada?	Conteúdo
AUTH_SRID	INTEGER	Sim	O identificador que a organização especificada na coluna AUTH_NAME assinala para este sistema de referência espacial.
SRTEXT	VARCHAR(2048)	Não	Texto de anotação para este sistema de referência espacial.
FALSEX	FLOAT	Não	Um número que, quando subtraído de um valor negativo da coordenada X, deixa um número não negativo (ou seja, um número positivo ou zero).
FALSEY	FLOAT	Não	Um número que, quando subtraído de um valor negativo da coordenada Y, permite um número não negativo (ou seja, um número positivo ou zero).
XYUNITS	FLOAT	Não	Um número que quando multiplicado por uma coordenada X decimal ou por uma coordenada Y decimal produz um inteiro que pode ser armazenado como um item de dados de 32–bits.
FALSEZ	FLOAT	Não	Um número que, quando subtraído de um valor negativo da coordenada Z, permite um número não negativo (ou seja, um número positivo ou zero).
ZUNITS	FLOAT	Não	Um número que quando multiplicado por uma coordenada Z decimal produz um inteiro que pode ser armazenado como um item de dados de 32–bits.
FALSEM	FLOAT	Não	Um número que, quando subtraído de uma medida negativa, permite um número não negativo (ou seja, um número positivo ou zero).
MUNITS	FLOAT	Não	Um número que quando multiplicado por uma medida decimal produz um inteiro que pode ser armazenado como um item de dados de 32–bits.

Capítulo 16. Comandos do DB2 Spatial Extender

Use estes comandos para configurar o DB2 Spatial Extender e desenvolver projetos que usam dados espaciais.

Chamando Comandos para Configurar o DB2 Spatial Extender e Desenvolver Projetos

Use o processador de linha de comandos (CLP) do DB2 Spatial Extender, chamado db2se, para configurar o Spatial Extender e criar projetos que usam dados espaciais. Este tópico explica como usar o **db2se** para executar comandos do DB2 Spatial Extender.

Pré-Requisitos

Antes de emitir comandos **db2se**, você deve estar autorizado para isso. Para saber qual autorização é necessária para um determinado comando, consulte a seção Autorização em cada comando.

Insira comandos **db2se** a partir de um prompt do sistema operacional.

Para saber quais comandos db2se e parâmetros estão disponíveis:

- Digite `db2se` ou `db2se -h`; em seguida, pressione Enter. É exibida uma lista de subcomandos do db2se.
- Digite `db2se` e um comando, ou `db2se` e um comando seguido pelo parâmetro **-h**. Em seguida, pressione Enter. É exibida a sintaxe necessária para o subcomando. Nesta sintaxe:
  - Cada parâmetro é precedido por um traço e seguido por um marcador para um valor de parâmetro.
  - Os parâmetros entre colchetes são opcionais. Os outros parâmetros são obrigatórios.

Para emitir um comando do db2se, digite `db2se`. Em seguida, digite um comando seguido pelos parâmetros e valores de parâmetro requeridos pelo comando db2se. Por último, pressione Enter.

É necessário digitar o ID do usuário e senha que lhe concedem acesso ao banco de dados que acabou de ser especificado. Por exemplo, digite o ID do usuário e senha, se desejar conectar-se ao banco de dados como um usuário que não seja você mesmo. Sempre preceda o ID do usuário com o parâmetro **-userId** e preceda a senha com o parâmetro **-pw**. Se você não especificar um ID do usuário e senha, seu ID do usuário e senha atuais serão utilizados por padrão.

Por padrão, os valores inseridos não fazem distinção entre maiúsculas e minúsculas. Para que eles façam distinção entre maiúsculas e minúsculas, coloque-os entre aspas duplas. Por exemplo, para especificar o nome de tabela mytable em minúsculas digite o seguinte: "mytable".

Pode ser necessário incluir escape nas aspas para assegurar que elas não sejam interpretadas como o prompt do sistema (shell), por exemplo, especifique \"mytable\" para referir-se à tabela chamada mytable. Se um valor com distinção

entre maiúsculas e minúsculas for qualificado por outro valor com distinção entre maiúsculas e minúsculas, delimite os dois valores individualmente; por exemplo, "myschema"."mytable". Coloque as sequências entre aspas duplas, da seguinte forma: "select * from newtable"

Importante: Não coloque toda a lista de nomes e valores de parâmetros entre aspas duplas.

A maioria dos comandos **db2se** executam um procedimento armazenado correspondente no servidor DB2 com exceção do comando **db2se shape_info**. Este comando é executado no cliente e pode acessar informações de coordenadas e do sistema de referência espacial de um banco de dados espacial ativado.

Os comandos **db2se import_shape** e **db2se export_shape** também podem ser executados no cliente. Isto é conveniente, pois permite acesso a arquivos locais no cliente.

Comando db2se alter_cs

O comando **db2se alter_cs** atualiza uma definição do sistema de coordenadas.

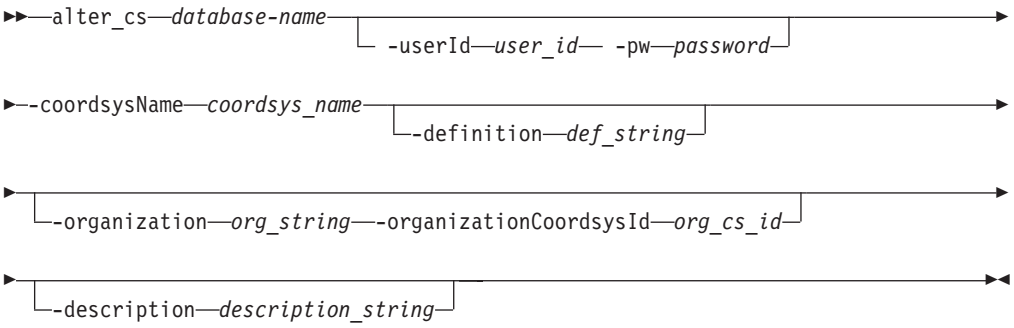
É possível utilizar este comando para atualizar a sequência de definição, o nome da organização, o identificador do sistema de coordenadas da organização e as descrição em um sistema coordenado definido pelo comando **db2se create_cs** ou pelo procedimento armazenado ST_CREATE_COORDSYS. As informações sobre o sistema de coordenadas estão disponíveis por meio da visualização de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Autorização

O ID do usuário deve ter as autoridades DBADM e DATAACCESS no banco de dados espacial ativado para executar este comando.

Sintaxe do comando

Comando db2se alter_cs



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja alterar o sistema coordenado.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-coordsysName *coordsys_name*
Identifica exclusivamente o sistema de coordenadas. O comprimento máximo para este parâmetro é de 128 caracteres.

O valor de *coordsys_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-definition *def_string*
Defina o sistema de coordenadas. O fornecedor do sistema de coordenadas geralmente possui as informações para este parâmetro. O comprimento máximo para este parâmetro é de 2048 caracteres.

-organization *org_string*
Define o nome da organização que definiu o sistema de coordenadas e forneceu a definição para ele; por exemplo, "European Petroleum Survey Group (EPSG)." O comprimento máximo para este parâmetro é de 128 caracteres.

A combinação de *org_string* e *org_cs_id* identifica exclusivamente o sistema de coordenadas.

-organizationCoordsysId *org_cs_id*
Especifica um identificador numérico. A entidade especificada em *org_string* designa este valor. O valor não é necessariamente exclusivo entre todos sistemas de coordenadas.

A combinação de *org_string* e *org_cs_id* identifica exclusivamente o sistema de coordenadas.

-description *description_string*
Descreve o sistema de coordenadas, explicando sua finalidade. O comprimento máximo para este parâmetro é de 256 caracteres.

Observações de Uso

Se usar este comando para alterar a definição do sistema de coordenadas e tiver dados espaciais existentes que estão associados a um sistema de referência espacial baseado neste sistema de coordenadas, você poderá alterar inadvertidamente os dados espaciais. Se os dados espaciais forem afetados, você será responsável por assegurar que os dados espaciais alterados ainda sejam precisos e válidos.

Exemplo

O exemplo a seguir atualiza a definição de um sistema de coordenadas chamado MYCOORDSYS com um novo nome da organização.

```
db2se alter_cs mydb -coordsysName mycoordsys -organization myNeworganizationb
```

Comando db2se alter_srs

O comando **db2se alter_srs** atualiza uma definição do sistema de referência espacial.

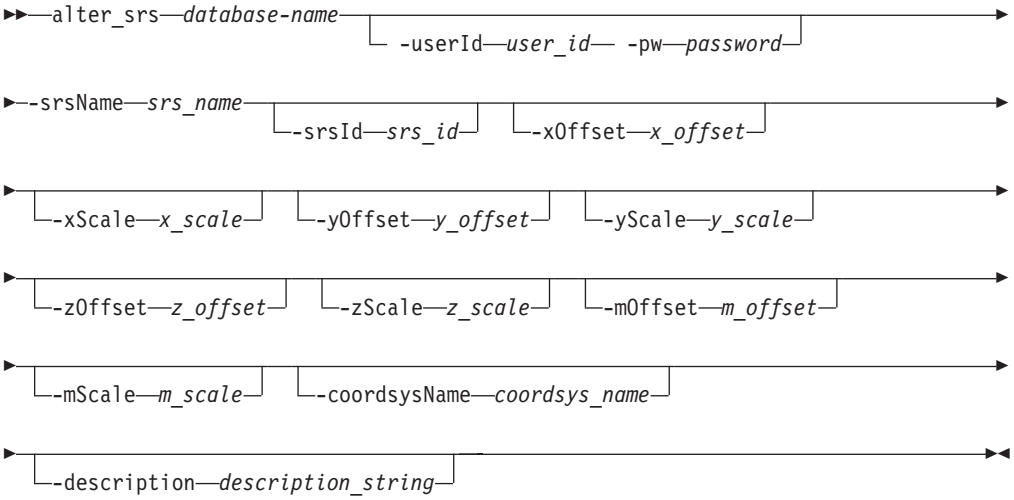
É possível usar este comando para alterar a compensação, escala ou nome do sistema de coordenadas de um sistema de referência espacial. As informações sobre o sistema de coordenadas estão disponíveis por meio da visualização de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Autorização

O ID do usuário deve ter as autoridades DBADM e DATAACCESS no banco de dados espacial ativado para executar este comando.

Sintaxe do comando

Comando db2se alter_srs



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja atualizar uma definição do sistema de referência espacial.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-srsName *srs_name*
Identifica o sistema de referência espacial que você deseja atualizar. O valor de *srs_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-srsId *srs_id*
Identifica exclusivamente o sistema de referência espacial. Este identificador é utilizado como um parâmetro de entrada para várias funções espaciais.

-xOffset *x_offset*
Especifica o deslocamento de todas as coordenadas X de geometrias que são representadas neste sistema de referência espacial. A compensação é subtraída

antes da aplicação do fator de escala *x_scale* quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como texto reconhecido (WKT), binário reconhecido (WKB) e forma.

-xScale *x_scale*
Especifica o fator de escala de todas as coordenadas X de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após a compensação *x_offset* ter sido subtraída quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-yOffset *y_offset*
Especifica o deslocamento de todas as coordenadas Y de geometrias que são representadas neste sistema de referência espacial. A compensação é subtraída antes da aplicação do fator de escala *y_scale* quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-yScale *y_scale*
Especifica o fator de escala de todas as coordenadas Y de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após a compensação *y_offset* ter sido subtraída quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma. Este fator de escala deve ser igual a *x_scale*.

-zOffset *z_offset*
Especifica o deslocamento de todas as coordenadas Z de geometrias que são representadas neste sistema de referência espacial. A compensação é subtraída antes da aplicação do fator de escala *z_scale* quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-zScale *z_scale*
Especifica o fator de escala de todas as coordenadas Z de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após a compensação *z_offset* ter sido subtraída quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-mOffset *m_offset*
Especifica o deslocamento de todas as coordenadas M de geometrias que são representadas neste sistema de referência espacial. A compensação é subtraída antes da aplicação do fator de escala *m_scale* quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-mScale *m_scale*
Especifica o fator de escala de todas as coordenadas M de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após a compensação *m_offset* ter sido subtraída quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-coordsysName *coordsys_name*
Identifica exclusivamente o sistema de coordenadas no qual se baseia este sistema de referência espacial. O sistema de coordenadas deve ser listado na visualização DB2GSE.ST_COORDINATE_SYSTEMS.

O valor de *coordsys_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas. O comprimento máximo para este parâmetro é de 128 caracteres.

-description *description_string*
Descreve o sistema de coordenadas, explicando sua finalidade. O comprimento máximo para este parâmetro é de 256 caracteres.

Observações de Uso

Se usar este comando para alterar a compensação, escala ou parâmetros coordsys_name do sistema de referência espacial, e se tiver dados espaciais existentes que estão associados ao sistema de referência espacial, você poderá alterar inadvertidamente os dados espaciais. Se os dados espaciais forem afetados, você será responsável por assegurar que os dados espaciais alterados ainda sejam precisos e válidos.

Exemplo

O exemplo a seguir altera um sistema de referência espacial chamado MYSRS com uma descrição diferente.

```
db2se alter_srs mydb -srsName mysrs -description "This is my own spatial reference system."
```

O exemplo a seguir altera um sistema de referência espacial chamado MYSRS_35 com um xOffset e descrição diferentes, porque não há dados espaciais que usam MYSRS_35. Se você tiver dados espaciais usando-o, os dados ficarão inutilizáveis.

```
db2se alter_srs mydb -srsName mysrs_35 -xOffset 35
        -description "This is my own spatial reference system with xOffset=35."
```

Comando db2se create_cs

O comando **db2se create_cs** cria um sistema de coordenadas.

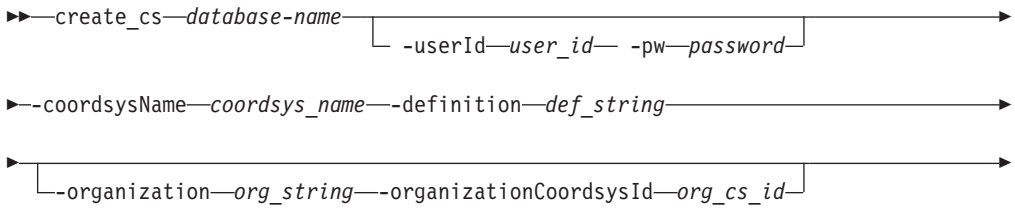
Este comando armazena informações no banco de dados sobre um novo sistema de coordenadas. As informações sobre o sistema de coordenadas estão disponíveis por meio da visualização de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Autorização

O ID do usuário deve ter as autoridades DBADM e DATAACCESS no banco de dados espacial ativado para executar este comando.

Sintaxe do comando

Comando db2se create_cs



Parâmetros de Comando

Em que:

database-name
O nome do banco de dados para o qual você deseja criar o sistema de coordenadas.

-userId *user_id*
O ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
A senha para *user_id*.

-coordsysName *coordsys_name*
Identifica exclusivamente o sistema de coordenadas. O comprimento máximo para este parâmetro é 128.

O valor de *coordsys_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-definition *def_string*
Defina o sistema de coordenadas. O fornecedor do sistema de coordenadas geralmente possui as informações para este parâmetro. O comprimento máximo para este parâmetro é 2048.

-organization *org_string*
Define o nome da organização que definiu o sistema de coordenadas e forneceu a definição para ele; por exemplo, "European Petroleum Survey Group (EPSG)."

A combinação de *org_string* e *org_cs_id* identifica exclusivamente o sistema de coordenadas. O comprimento máximo para este parâmetro é 128.

-organizationCoordsysId *org_cs_id*
Especifica um identificador numérico. A entidade especificada em *org_string* designa este valor. O valor não é necessariamente exclusivo entre todos sistemas de coordenadas.

A combinação de *org_string* e *org_cs_id* identifica exclusivamente o sistema de coordenadas.

-description *description_string*
Descreve o sistema de coordenadas, explicando sua finalidade. O comprimento máximo para este parâmetro é 256.

Observações de Uso

O DB2 Spatial Extender fornece mais de 4000 sistemas de coordenadas. Criar um sistema de coordenadas é raramente necessário. Além disso, você deve criar um sistema de referência espacial baseado no novo sistema de coordenadas.

Exemplo

O exemplo a seguir cria um sistema de coordenadas chamado MYCOORDSYS.

```
db2se create_cs mydb -coordsysName mycoordsys
                     -definition GEOCS[\"GCS_NORTH_AMERICAN_1983\",
                                 DATUM["D_North_American_1983\",
                                 SPHEROID[\"GRS_1980\",6387137,298.257222101]],
                                 PRIMEM[\"Greenwich\",0],
                                 UNIT["Degree\", 0.0174532925199432955]]
```

Comando db2se create_srs

O comando **db2se create_srs** cria um sistema de referência espacial.

É possível usar este comando para criar uma definição do sistema de referência espacial. Um sistema de referência espacial é definido pelo sistema de coordenadas, pela precisão e as extensões de coordenadas que são representadas no sistema de referência espacial especificado. As extensões são os valores de coordenadas mínimo e máximo para as coordenadas X, Y, Z e M. As informações sobre o sistema de coordenadas estão disponíveis por meio da visualização de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Este comando possui dois métodos para criar sistemas de referência espacial:

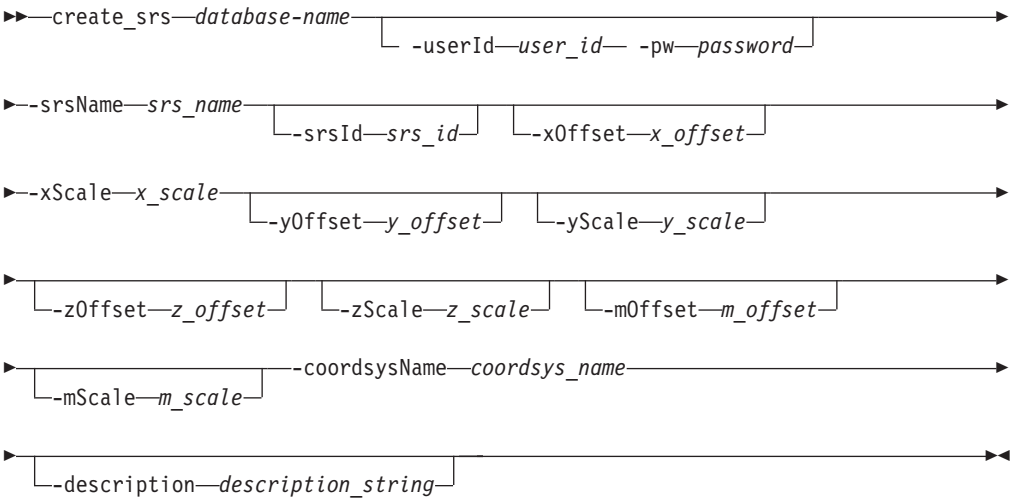
- Usando fatores de conversão (compensações e fatores de escala)
- Usando extensões e precisão (os fatores de conversão são calculados)

Autorização

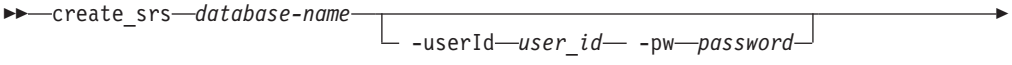
O ID do usuário deve ter as autoridades DBADM e DATAACCESS no banco de dados espacial ativado para executar este comando.

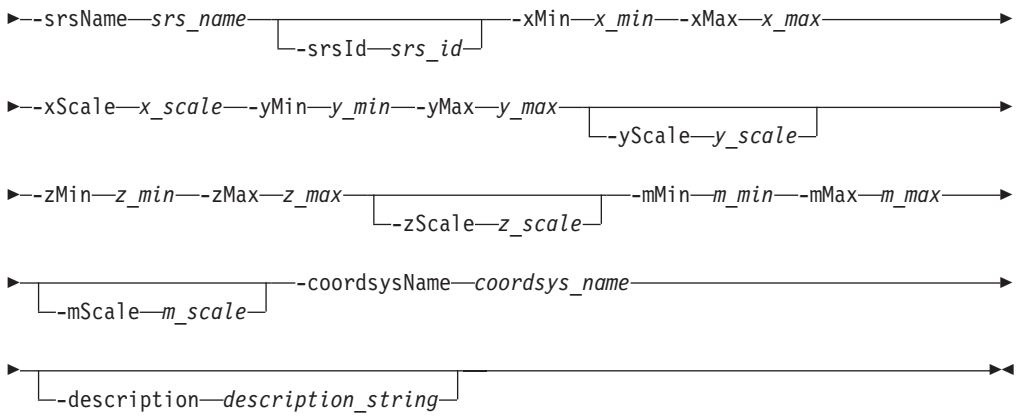
Sintaxe do comando

Comando db2se create_srs (Usando Fatores de Conversão)



Comando db2se create_srs (Usando Extensões e Precisão)





Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja criar uma definição do sistema de referência espacial.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-srsName *srs_name*
Identifica o sistema de referência espacial que você deseja criar. O valor de *srs_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-srsId *srs_id*
Identifica exclusivamente o sistema de referência espacial. Este identificador é utilizado como um parâmetro de entrada para várias funções espaciais.

-xMin *x_min*
Especifica o valor de coordenadas X mínimo para todas as geometrias que usam o sistema de referência espacial especificado.

-xMax *x_max*
Especifica o valor de coordenadas X máximo para todas as geometrias que usam o sistema de referência espacial especificado. Dependendo do valor *x_scale*, o valor de coordenadas X máximo retornado pela visualização DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS pode ser maior que o valor especificado por *x_max*. O valor retornado pela visualização está correto.

-xOffset *x_offset*
Especifica a compensação para todas as coordenadas X de geometrias que são representadas no sistema de referência espacial especificado. A compensação é subtraída antes da aplicação do fator de escala *x_scale* quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como texto reconhecido (WKT), binário reconhecido (WKB) e forma.

-xScale *x_scale*
Especifica o fator de escala para todas as coordenadas X de geometrias que são representadas no sistema de referência espacial especificado. O fator de escala

é aplicado (multiplicação) após a compensação *x_offset* ter sido subtraída quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-yMin *y_min*
Especifica o valor de coordenadas Y mínimo para todas as geometrias que usam o sistema de referência espacial especificado.

-yMax *y_max*
Especifica o valor de coordenadas Y máximo para todas as geometrias que usam o sistema de referência espacial especificado. Dependendo do valor *y_scale*, o valor máximo de coordenadas Y retornado pela visualização DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS pode ser maior do que o valor *y_max* especificado. O valor retornado pela visualização está correto.

-yOffset *y_offset*
Especifica a compensação para todas as coordenadas Y de geometrias que são representadas no sistema de referência espacial especificado. A compensação é subtraída antes da aplicação do fator de escala *y_scale* quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-yScale *y_scale*
Especifica o fator de escala para todas as coordenadas Y de geometrias que são representadas no sistema de referência espacial especificado. O fator de escala é aplicado (multiplicação) após a compensação *y_offset* ter sido subtraída quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma. Este fator de escala deve ser igual a *x_scale*.

-zMin *z_min*
Especifica o valor de coordenadas Z mínimo para todas as geometrias que usam o sistema de referência espacial especificado.

-zMax *z_max*
Especifica o valor de coordenadas Z máximo para todas as geometrias que usam o sistema de referência espacial especificado. Dependendo do valor *z_scale*, o valor máximo de coordenadas Z retornado pela visualização DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS pode ser maior do que o valor *z_max* especificado. O valor retornado pela visualização está correto.

-zOffset *z_offset*
Especifica a compensação para todas as coordenadas Z de geometrias que são representadas no sistema de referência espacial especificado. A compensação é subtraída antes da aplicação do fator de escala *z_scale* quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-zScale *z_scale*
Especifica o fator de escala para todas as coordenadas Z de geometrias que são representadas no sistema de referência espacial especificado. O fator de escala é aplicado (multiplicação) após a compensação *z_offset* ter sido subtraída quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-mMin *m_min*
Especifica o valor de coordenadas M mínimo para todas as geometrias que usam o sistema de referência espacial especificado.

-mMax *m_max*
Especifica o valor de coordenadas M máximo para todas as geometrias que

usam o sistema de referência espacial especificado. Dependendo do valor *m_scale*, o valor de coordenadas M máximo retornado pela visualização DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS pode ser maior que o valor *m_max* especificado. O valor retornado pela visualização está correto.

-mOffset *m_offset*
Especifica a compensação para todas as coordenadas M de geometrias que são representadas no sistema de referência espacial especificado. A compensação é subtraída antes da aplicação do fator de escala *m_scale* quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-mScale *m_scale*
Especifica o fator de escala para todas as coordenadas M de geometrias que são representadas no sistema de referência espacial especificado. O fator de escala é aplicado (multiplicação) após a compensação *m_offset* ter sido subtraída quando as geometrias são convertidas na representação interna do DB2 Spatial Extender a partir de representações externas, tais como, WKT, WKB e forma.

-coordsysName *coordsys_name*
Identifica exclusivamente o sistema de coordenadas no qual o sistema de referência espacial é baseado. O sistema de coordenadas deve ser listado na visualização DB2GSE.ST_COORDINATE_SYSTEMS.

O valor de *coordsys_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas. O comprimento máximo para este parâmetro é 128.

-description *description_string*
Descreve o sistema de coordenadas, explicando sua finalidade. O comprimento máximo para este parâmetro é 256.

Observações de Uso

O DB2 Spatial Extender inclui suporte para cinco sistemas de referência espacial e mais de 4000 sistemas de coordenadas dos quais é possível escolher. Raramente é necessária a criação de um sistema de referência espacial.

Exemplo

O exemplo a seguir cria um sistema de referência espacial chamado MYSRS usando a sintaxe de comando para fatores de conversão.

```
db2se create_srs mydb -srsName mysrs -srsID 100
        -xOffset -180 -xScale 1000000 -yOffset -90
        -coordsysName \"GCS_North_American_1983\"
```

Comando db2se disable_autogc

O comando **db2se disable_autogc** desativa a geocodificação automática.

Este comando impede que o DB2 Spatial Extender sincronize uma coluna geocodificada com sua coluna ou colunas geocodificadas associadas. Uma coluna de codificação geográfica é usada como entrada para o geocodificador. Toda vez que os valores são inseridos ou atualizados na coluna ou colunas de codificação geográfica, os disparos são ativados. Esses disparos chamam o geocodificador associado para geocodificar os valores inseridos ou atualizados e para colocar os

dados resultantes na coluna codificada geograficamente. As informações sobre colunas geocodificadas estão disponíveis na visualização de catálogo DB2GSE.ST_GEOCODING.

Autorização

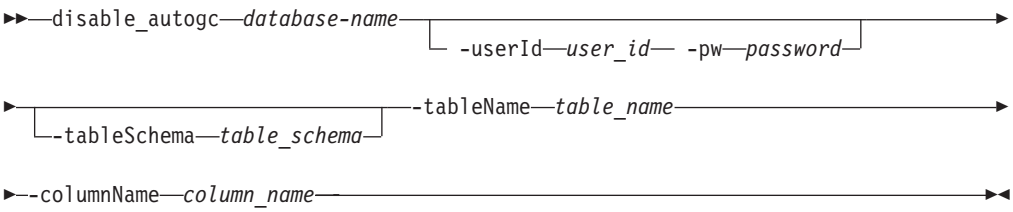
O ID do usuário deve ter uma das seguintes autoridades ou privilégios para executar este comando:

- São definidas as autoridades DBADM e DATAACCESS no banco de dados que contém a tabela na qual os acionadores que estão sendo eliminados.
- Privilégio CONTROL nesta tabela e autoridade DROPIN no esquema DB2GSE.
- Privilégios ALTER e UPDATE nesta tabela e autoridade DROPIN no esquema DB2GSE.

Além disso, o ID do usuário deve ter os privilégios necessários no servidor DB2 para criar ou gravar nos arquivos de exceção e de mensagens.

Sintaxe do comando

Comando db2se disable_autogc



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja desativar a geocodificação automática.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-tableSchema *table_schema*
Especifica o nome do esquema para o *table_name* especificado. Se você não especificar um nome do esquema, o valor no registro especial CURRENT SCHEMA será usado como o nome do esquema para a tabela ou visualização.

-tableName *table_name*
Especifica o nome não qualificado da tabela contendo a coluna especificada na qual você deseja desativar a geocodificação automática. Os acionadores criados para sincronizar a coluna geocodificada são eliminados. O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-columnName *column_name*
Especifica os nomes da coluna geocodificada mantida na qual você deseja

desativar a geocodificação automática. O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

Exemplo

O exemplo a seguir desativa a codificação geográfica automática de uma coluna com codificação geográfica denominada MYCOLUMN na tabela MYTABLE.

```
db2se disable_autogc mydb -tableName mytable -columnName mycolumn
```

Comando db2se disable_db

O comando **db2se disable_db** remove recursos que permitem que o DB2 Spatial Extender armazene e suporte dados espaciais.

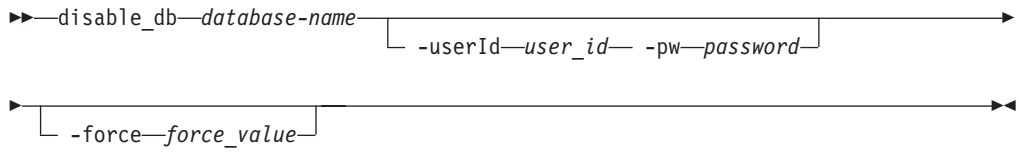
Estes recursos incluem tipos de dados espaciais, tipos de índices espaciais, visualizações de catálogo, funções fornecidas e procedimentos armazenados que foram criados durante a ativação de um banco de dados para suporte de operações espaciais.

Autorização

O ID do usuário deve ter as autoridades DBADM e DATAACCESS no banco de dados espacial ativado para executar este comando.

Sintaxe do comando

Comando db2se disable_db



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja desativar o DB2 Spatial Extender.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-force *force_value*
Especifica que você deseja desativar um banco de dados para operações espaciais, mesmo que você tenha objetos de banco de dados que sejam dependentes dos tipos espaciais ou funções espaciais. Os objetos de banco de dados que podem ter essas dependências incluem tabelas, exibições, limitações, disparos, colunas geradas, métodos, funções, procedimentos e outros tipos de dados (subtipos ou tipos estruturados com um atributo espacial).

Se você especificar um valor diferente de zero para o parâmetro **-force**, o banco de dados será desativado e todos os recursos do DB2 Spatial Extender serão removidos. Se você especificar zero como *force_value*, o banco de dados será desativado apenas se os objetos de banco de dados forem independentes de tipos espaciais ou de funções espaciais.

Observações de Uso

Se você já definiu colunas espaciais mas ainda deseja desativar um banco de dados para operações espaciais, deverá especificar um valor diferente de 0 (zero) para o parâmetro force para remover todos os recursos espaciais no banco de dados que não possuem outras dependências deles.

Exemplo

O exemplo a seguir desativa o suporte para operações espaciais no banco de dados *MYDB*, mesmo que haja objetos de banco de dados com dependências de tipos e funções espaciais.

```
db2se disable_db mydb -force 1
```

Comando db2se drop_cs

O comando **db2se drop_cs** exclui uma definição do sistema de coordenadas.

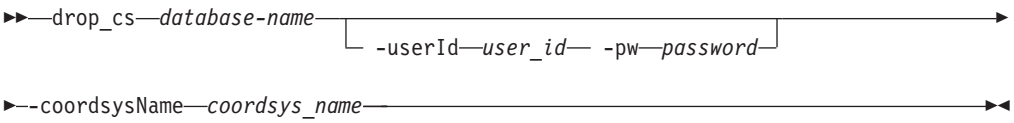
Este comando exclui informações sobre um sistema de coordenadas do banco de dados. As informações não estão mais disponíveis por meio da visualização de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Autorização

O ID do usuário deve ter as autoridades DBADM e DATAACCESS no banco de dados espacial ativado para executar este comando.

Sintaxe do comando

Comando db2se drop_cs



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja excluir o sistema de coordenadas.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para o *user_id* especificado.

-coordsysName *coordsys_name*
Identifica exclusivamente o sistema de coordenadas que você deseja excluir.

O valor de *coordsys_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas. O comprimento máximo para este parâmetro é 128.

Observações de Uso

Não é possível eliminar um sistema de coordenadas no qual baseia-se um sistema de referência espacial.

Exemplo

O exemplo a seguir elimina um sistema de coordenadas chamado MYCOORDSYS.

```
db2se drop_cs mydb -coordsysName mycoordsys
```

Comando db2se drop_srs

O comando **db2se drop_srs** exclui uma definição do sistema de referência espacial.

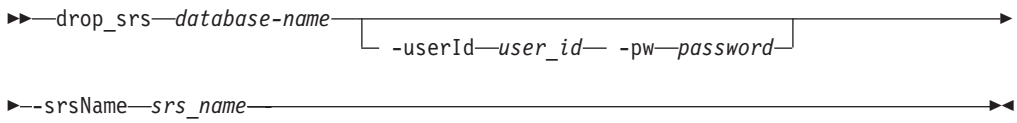
As informações sobre o sistema de referência espacial não estão mais disponíveis por meio da visualização de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Autorização

O ID do usuário possui as autoridades DBADM e DATAACCESS no banco de dados espacial ativado para executar este comando.

Sintaxe do comando

Comando db2se drop_srs



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja excluir uma definição do sistema de referência espacial.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-srsName *srs_name*
Identifica o sistema de referência espacial que você deseja excluir. O valor de *srs_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

Observações de Uso

Se você registrou colunas espaciais que usam um sistema de referência espacial, não será possível eliminar esse sistema de referência espacial.

Exemplo

O exemplo a seguir elimina um sistema de referência espacial chamado MYSRS.

```
db2se drop_srs mydb -srsName mysrs
```

Comando db2se enable_autogc

O comando **db2se enable_autogc** ativa a geocodificação automática.

Este comando permite que o DB2 Spatial Extender sincronize uma coluna geocodificada com sua coluna ou colunas de codificação geográfica associadas. Uma coluna de codificação geográfica é usada como entrada para o geocodificador. Toda vez que os valores são inseridos ou atualizados na coluna ou colunas de codificação geográfica, os disparos são ativados. Esses disparos chamam o geocodificador associado para geocodificar os valores inseridos ou atualizados e para colocar os dados resultantes na coluna codificada geograficamente. As informações sobre colunas geocodificadas estão disponíveis na visualização de catálogo DB2GSE.ST_GEOCODING.

Autorização

O ID do usuário deve ter uma das seguintes autoridades ou privilégios para executar este comando:

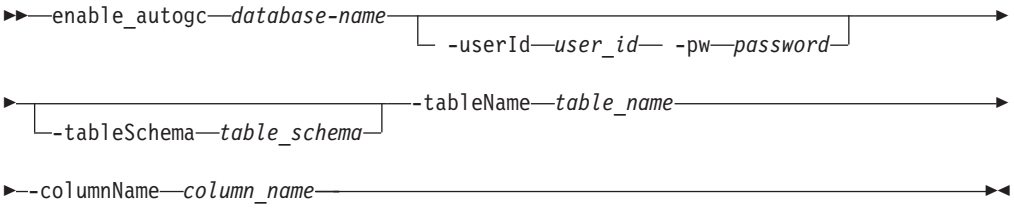
- Autoridades DBADM e DATAACCESS no banco de dados que contém a tabela na qual os acionadores criados por este procedimento armazenado são definidos
- Privilégio CONTROL sobre a tabela
- Privilégio ALTER na tabela

Se o ID de autorização da instrução não tiver autoridade DBADM, os privilégios que o ID de autorização da instrução mantém (sem considerar os privilégios PUBLIC ou de grupo) deverão incluir todos os seguintes privilégios, desde que exista o acionador:

- Privilégio SELECT na tabela na qual a geocodificação automática está ativada ou a autoridade DATAACCESS.
- Privilégios necessários para avaliar as expressões SQL especificadas para os parâmetros na configuração de codificação geográfica.

Sintaxe do comando

Comando db2se enable_autogc



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja ativar a geocodificação automática.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-tableSchema *table_schema*
Especifica o nome do esquema para o *table_name* especificado. Se você não especificar um nome do esquema, o valor no registro especial CURRENT SCHEMA será usado como o nome do esquema para a tabela ou visualização.

-tableName *table_name*
Especifica o nome não qualificado da tabela que contém a coluna especificada na qual você deseja ativar a geocodificação automática. Os acionadores são criados para sincronizar a coluna geocodificada. O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-columnName *column_name*
Identifica a coluna na qual os dados geocodificados serão inseridos ou atualizados. Esta coluna é chamada de coluna geocodificada. O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

Observações de Uso

Antes de ativar a geocodificação automática, você deve executar a etapa de configuração de codificação geográfica para especificar o geocodificador e os valores de parâmetro de codificação geográfica. Ela também identifica as colunas de codificação geográfica que devem ser sincronizadas com as colunas geocodificadas.

Você só pode ativar a geocodificação automática nas tabelas nas quais os acionadores INSERT e UPDATE podem ser criados. Consequentemente, não é possível ativar o autogeocoding nas exibições ou nos pseudônimos.

Exemplo

O exemplo a seguir configura a codificação geográfica automática de uma coluna denominada MYCOLUMN na tabela MYTABLE.

```
db2se enable_autogeocoding mydb -tableName mytable -columnName mycolumn
```

Comando db2se enable_db

O comando **db2se enable_db** fornece a um banco de dados os recursos de que ele precisa para armazenar dados espaciais e suportar operações espaciais.

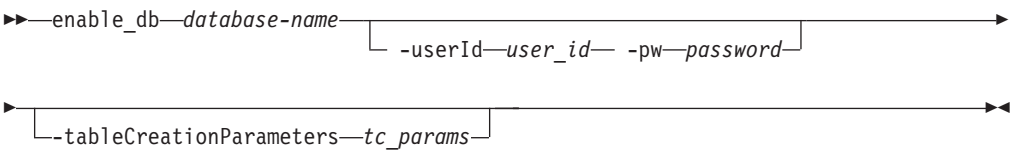
Estes recursos incluem tipos de dados espaciais, tipos de índices espaciais, exibições do catálogo, funções fornecidas e outros procedimentos armazenados.

Autorização

O ID do usuário deve ter as autoridades DBADM, DATAACCESS e CTRLACCESS no banco de dados espacial ativado para executar este comando.

Sintaxe do comando

Comando db2se enable_db



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual deseja ativar o DB2 Spatial Extender.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-tableCreationParameters *tc_params*
Especifica parâmetros para as instruções CREATE TABLE que são usadas para criar tabelas de catálogos do DB2 Spatial Extender. Use a sintaxe do parâmetro para a instrução CREATE TABLE. O exemplo a seguir no sistema operacional Windows especifica um espaço de tabela no qual criar as tabelas e um espaço de tabela no qual criar os índices da tabela:

```
-tableCreationParameters "IN TS_name INDEX IN IDX_TS_name"
```

O comprimento máximo para este parâmetro é de 32.672 caracteres.

Observações de Uso

Certifique-se de que você tenha um espaço de tabela temporário do sistema com um tamanho de página de 8 KB ou maior e com um tamanho mínimo de 500 páginas. Isso é um requisito para executar o **db2se enable_db** com êxito.

Exemplo

O exemplo a seguir ativa um banco de dados chamado MYDB para operações espaciais.

```
db2se enable_db mydb
```

Comando db2se export_shape

O comando **db2se export_shape** exporta uma coluna espacial e sua tabela associada para um arquivo de forma.

É possível usar este comando para criar uma definição do sistema de referência espacial. Um sistema de referência espacial é definido pelo sistema de coordenadas, pela precisão e as extensões de coordenadas que são representadas no sistema de referência espacial especificado. As extensões são os valores de coordenadas mínimo e máximo possíveis para as coordenadas X, Y, Z e M. As informações sobre o sistema de coordenadas estão disponíveis por meio da visualização de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

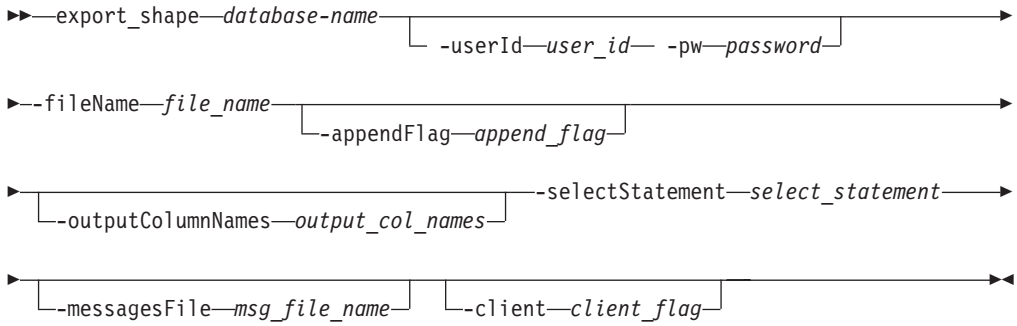
Autorização

O usuário deve ter os privilégios necessários para executar com êxito a instrução SELECT da qual os dados devem ser exportados.

Além disso, o ID do proprietário da instância do DB2 deve ter os privilégios necessários no servidor DB2 para criar ou gravar nos arquivos de forma e no arquivo de mensagens.

Sintaxe do comando

Comando db2se export_shape



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados que contém a tabela a ser exportada.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-fileName *file_name*
Especifica o nome completo do caminho de um arquivo shape para o qual os dados especificados serão exportados. Consulte “Observações de Uso” na página 147 para obter uma lista completa de arquivos que são gravados no servidor DB2. O comprimento máximo para este parâmetro é 256.

Se você estiver exportando para um novo arquivo, poderá especificar a extensão do arquivo opcional como .shp ou .SHP. Se você especificar .shp ou .SHP como a extensão do arquivo, o DB2 Spatial Extender criará o arquivo com o valor *file_name* especificado. Se você não especificar a extensão do arquivo opcional, o DB2 Spatial Extender criará o arquivo com o nome *file_name*.shp.

Se estiver exportando dados anexando-os a um arquivo existente, o DB2 Spatial Extender primeiro procurará uma correspondência exata do nome especificado com o parâmetro **-fileName**. Se o DB2 Spatial Extender não encontrar uma correspondência exata, ele primeiro procurará um arquivo com a extensão .shp e, em seguida, um arquivo com a extensão .SHP. Se o valor *append_flag* indicar que você não está anexando a um arquivo existente, mas o arquivo já existir, o DB2 Spatial Extender retornará um erro e não sobrescreverá o arquivo.

-appendFlag *append_flag*
Indica se os dados que devem ser exportados serão anexados a um arquivo shape existente. Os possíveis valores para este parâmetro são:

- Um valor diferente de zero em *append_flag* para indicar que você deseja anexar dados a um arquivo de forma existente. Se a estrutura do arquivo existente não corresponder aos dados exportados, será retornado um erro.
- Um valor de 0 em *append_flag* para indicar que você deseja exportar para um novo arquivo. O DB2 Spatial Extender não sobrescreve arquivos existentes.

-outputColumnNames *output_col_names*
Especifica um ou mais nomes de colunas (separados por vírgulas) que são usados para colunas não espaciais no arquivo dBASE de saída. Se este parâmetro não for especificado, os nomes de colunas da instrução SELECT serão usados.

Se os nomes de colunas não forem colocados entre aspas duplas, eles serão convertidos em maiúsculas. O número de colunas especificadas deve corresponder ao número de colunas que são retornadas da instrução SELECT indicada no parâmetro *select_statement*, excluindo a coluna espacial.

O comprimento máximo para este parâmetro é de 32.672 caracteres.

-selectStatement *select_statement*
Especifica a subseleção que retorna os dados a serem exportados. A subseleção deve referir-se exatamente a uma coluna espacial e a qualquer número de colunas de atributo. O comprimento máximo para este parâmetro é de 32.672 caracteres.

-messagesFile *msg_file_name*
Especifica o nome do caminho completo do arquivo no servidor DB2 no qual o DB2 Spatial Extender grava mensagens sobre a operação de exportação. Se você especificar este parâmetro e o arquivo já existir, será retornado um erro e a operação de exportação será finalizada. Se você não especificar este parâmetro, o DB2 Spatial Extender não criará um arquivo de mensagens.

Os seguintes tipos de mensagens são gravados no arquivo de mensagens:

- Mensagens informativas como, por exemplo, um resumo da operação de exportação
- Mensagens de erro de dados que não puderam ser exportados, por exemplo, por causa de sistemas de coordenadas diferentes

O comprimento máximo para este parâmetro é 256.

-client *client_flag*
Especifica se a operação de exportação ocorre no cliente ou no servidor DB2 e onde os arquivos são criados. Os possíveis valores para este parâmetro são:

- 0 para indicar que a operação de exportação ocorre no servidor DB2 e os arquivos são criados no servidor DB2.
- 1 para indicar que a operação de exportação ocorre no cliente e os arquivos são criados no cliente.

Se você não especificar este parâmetro, o valor 0 será o padrão.

Observações de Uso

Você pode exportar apenas uma coluna espacial por vez.

É possível executar o processo de exportação no cliente em que o comando é executado. Isto geralmente é mais conveniente já que não requer acesso ao sistema de arquivos do servidor DB2.

O **db2se export_shape** cria ou grava nos quatro arquivos a seguir:

- O arquivo shape principal (extensão .shp).
- O arquivo shape de índice (extensão .shx).
- Um arquivo dBASE que contém dados de colunas não espaciais (extensão .dbf). Este arquivo é criado apenas se as colunas de atributo realmente precisarem ser exportadas
- Um arquivo de projeção que especifica o sistema de coordenadas que está associado aos dados espaciais, se o sistema de coordenadas não for igual a "UNSPECIFIED" (extensão .prj). O sistema de coordenadas é obtido do primeiro registro espacial. Ocorrerá um erro se registros subsequentes tiverem sistemas de coordenadas diferentes.

A tabela a seguir descreve como os tipos de dados DB2 são armazenados nos arquivos de atributos do dBASE. Todos os outros tipos de dados DB2 não são suportados.

Tabela 20. Armazenamento de Tipos de Dados DB2 nos Arquivos de Atributos

Tipo de SQL	Tipo de .dbf	Comprimento do .dbf	Decimais do .dbf	Comentários
SMALLINT	N	6	0	
INTEGER	N	11	0	
BIGINT	N	20	0	
DECIMAL	N	precision+2	escala	
REAL FLOAT(1) até FLOAT(24)	F	14	6	
DOUBLE FLOAT(25) até FLOAT(53)	F	19	9	
CHARACTER, VARCHAR, LONG VARCHAR e DATALINK	C	*len*	0	comprimento ≤ 255
DATE	D	8	0	
TIME	C	8	0	
TIMESTAMP	C	26	0	

Todos os sinônimos de tipos de dados e tipos distintos que baseiam-se nos tipos listados na tabela precedente são suportados.

Exemplo

O exemplo a seguir exporta uma coluna espacial chamada MYCOLUMN e sua tabela associada, MYTABLE, para o arquivo de forma `myshapefile` que está localizado no cliente.

```
db2se export_shape mydb -fileName /home/myaccount/myshapefile
        -selectStatement "select * from mytable" -client 1
```

Comando db2se import_shape

O comando **db2se import_shape** importa um arquivo de forma para um banco de dados ativado para operações espaciais.

Este comando pode importar dados de forma e de atributo para uma tabela existente ou uma nova tabela.

Autorização

O ID do proprietário da instância do DB2 deve ter os privilégios necessários no servidor DB2 para criar ou gravar nos arquivos de exceções e de mensagens.

O ID do usuário deve ter requisitos de autorização adicionais para executar este comando. Os requisitos podem variar, caso você esteja importando para uma tabela existente ou para uma nova tabela.

Requisitos para importar para uma tabela existente

O ID do usuário deve conter uma das seguintes autoridades ou privilégios:

- DATAACCESS
- Privilégio CONTROL na tabela ou visualização
- Privilégio INSERT e SELECT na tabela ou visualização

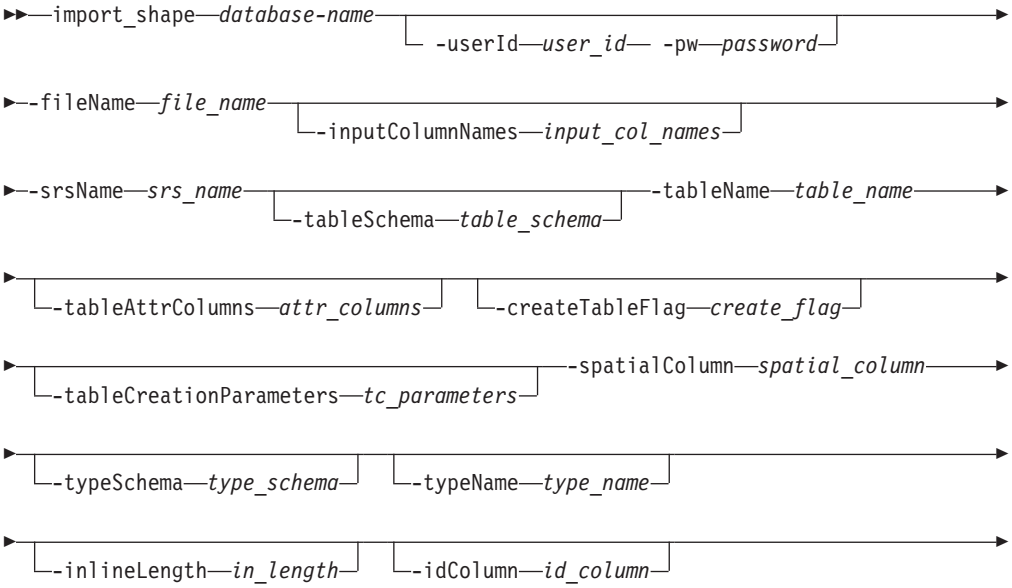
Requisitos para importar para uma nova tabela

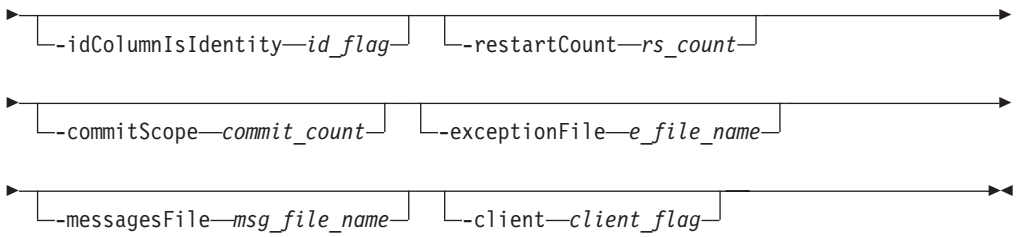
O ID do usuário deve conter uma das seguintes autoridades ou privilégios:

- DBADM e DATAACCESS
- Autoridade CREATETAB no banco de dados
- Autoridade IMPLICIT-SCHEMA no banco de dados, se o nome do esquema da tabela não existir
- Privilégio CREATEIN no esquema, se o esquema da tabela existir

Sintaxe do comando

Comando db2se import_shape





Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja importar o arquivo de forma.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-fileName *file_name*
Especifica o nome do caminho completo de um arquivo de forma para o qual o dados especificados devem ser importados. Se você especificar `.shp` ou `.SHP` como a extensão do arquivo, o DB2 Spatial Extender primeiro procurará uma correspondência exata do nome especificado com o parâmetro **-fileName**. Se o DB2 Spatial Extender não localizar uma correspondência exata, ele procurará primeiro um arquivo com a extensão `.shp` e, em seguida, um arquivo com a extensão `.SHP`. Consulte "Observações de Uso" na página 153 para obter uma lista completa de arquivos que são gravados no servidor DB2.

O comprimento máximo para este parâmetro é de 256 caracteres.

-inputColumnNames *input_col_names*
Especifica uma lista de colunas de atributo a serem importadas do arquivo dBASE. Se este parâmetro não for especificado, todas as colunas no arquivo serão importadas. Use qualquer um dos seguintes formatos para especificar uma lista de atributos:

- Uma lista separada por vírgula de nomes de colunas a serem importadas do arquivo dBASE conforme mostrado no exemplo a seguir:

  ```
  N(COLUMN1,COLUMN5,COLUMN3,COLUMN7)
  ```

  Se os nomes de colunas não forem colocados entre aspas duplas, eles serão convertidos em maiúsculas. Os nomes resultantes devem corresponder exatamente aos nomes de coluna no arquivo dBASE.
- Uma lista separada por vírgula de números de colunas a serem importadas do arquivo dBASE conforme mostrado no exemplo a seguir:

  ```
  P(1,5,3,7)
  ```

  As colunas são enumeradas iniciando com 1. Cada número na lista deve ser separado por uma vírgula.
- Uma sequência vazia `""` para indicar que nenhum dos dados do atributo deve ser importado.

O comprimento máximo para este parâmetro é de 32.672 caracteres.

-srsName *srs_name*
Identifica o sistema de referência espacial (SRS) a ser usado para as geometrias importadas para a coluna espacial. O valor de *srs_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

A coluna espacial não é registrada. O SRS deve existir antes da importação dos dados. O processo de importação não cria implicitamente o SRS, mas compara o sistema de coordenadas do SRS com o sistema de coordenadas especificado no arquivo `.prj` (se este arquivo estiver disponível com o arquivo de forma).

O processo de importação também verifica se as extensões dos dados no arquivo de forma podem ser representadas no SRS especificado. Ou seja, o processo de importação verifica se as extensões estão dentro das coordenadas X, Y, Z e M mínimas e máximas do SRS.

-tableSchema *table_schema*
Especifica o nome do esquema para o *table_name* especificado. Se você não especificar um nome do esquema, o valor no registro especial CURRENT SCHEMA será usado como o nome do esquema para a tabela ou visualização.

-tableName *table_name*
Especifica o nome não qualificado da tabela para a qual os dados no arquivo de forma devem ser importados. O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-tableAttrColumns *attr_columns*
Especifica os nomes de colunas da tabela nos quais os dados de atributos do arquivo dBASE serão armazenados. Se este parâmetro não for especificado, os nomes das colunas no arquivo dBASE serão usados.

O número de colunas especificadas deve corresponder ao número de colunas a serem importadas do arquivo dBASE. Se a tabela existir, as definições de colunas deverão corresponder aos dados de entrada. Consulte "Observações de Uso" na página 153 para obter uma explicação de como os tipos de dados do atributo são mapeados para tipos de dados DB2.

Se os nomes de colunas não forem colocados entre aspas duplas, eles serão convertidos em maiúsculas. O comprimento máximo para este parâmetro é de 32.672 caracteres.

-createTableFlag *create_flag*
Especifica se o processo de importação criará uma tabela. Os possíveis valores para este parâmetro são:

- Um valor diferente de zero em *create_flag* para criar uma tabela. Se a tabela existir, será retornado um erro.
- Um valor de 0 em *create_flag* para usar uma tabela existente.

Se este parâmetro não for especificado, será criada uma nova tabela.

-tableCreationParameters *tc_params*
Especifique opções que devem ser incluídas na instrução CREATE TABLE que cria o *table_name* especificado.

Para especificar opções CREATE TABLE, use a sintaxe da instrução CREATE TABLE. Por exemplo, para especificar um espaço de tabela no qual criar as tabelas, índices e objetos grandes, especifique *tc_params*:

```
IN tsName INDEX IN indexTsName LONG IN longTsName
```

O comprimento máximo para este parâmetro é de 32.672 caracteres.

-spatialColumn *spatial_column*
Especifica o nome da coluna espacial na tabela para a qual os dados de forma devem ser importados.

Para uma nova tabela, este parâmetro especifica o nome da nova coluna espacial que será criada. Para uma tabela existente, este parâmetro especifica o nome de uma coluna espacial existente na tabela.

O valor de *spatial_column* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-typeSchema *type_schema*
Indica o nome do esquema do tipo de dados espaciais especificado no valor *type_name*. Se este parâmetro não for especificado, DB2GSE será usado como o nome do esquema.

O valor de *type_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-typeName *type_name*
Especifica o nome do tipo de dados a ser usado para valores espaciais. Se este parâmetro não for especificado, o tipo de dados será determinado pelo arquivo de forma de qualquer um dos seguintes tipos de dados:

- ST_Point
- ST_MultiPoint
- ST_MultiLineString
- ST_MultiPolygon

Os arquivos de forma, por definição, permitem uma distinção apenas entre pontos e multipontos. Não há nenhuma distinção entre polígonos e multipolígonos ou entre sequências de linhas e sequências multilinhas.

Se estiver importando para uma nova tabela, o tipo de dados *type_name* também será usado para o tipo de dados da coluna espacial. Neste caso, o tipo de dados também pode ser um supertipo de ST_Point, ST_MultiPoint, ST_MultiLineString ou ST_MultiPolygon.

O valor de *type_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-inlineLength *inline_length*
Para uma nova tabela, especifica o número máximo de bytes que devem ser alocados para a coluna espacial na tabela. Se este parâmetro não for especificado, será usado o comprimento sequencial padrão.

Os registros espaciais que excedem o tamanho *inline_length* são armazenados separadamente no espaço de tabela de LOB, que pode ser mais lento para acessar.

Os tamanhos típicos que são necessários para vários tipos espaciais são os seguintes:

- **Um ponto:** 292 bytes.
- **Multiponto, linha ou polígono:** O maior valor possível. Considere que o número total de bytes em uma linha não deve exceder o limite do tamanho da página da área de tabela para a qual a tabela é criada.

Para obter uma descrição completa do valor *inline_length*, consulte a instrução CREATE TABLE na documentação do DB2. Use a função da tabela ADMIN_EST_INLINE_LENGTH para ajudá-lo a estimar o comprimento sequencial necessário para geometrias em tabelas existentes.

-idColumn *id_column*
Especifica o nome da coluna a ser criada para conter um número exclusivo para cada linha de dados. Os valores exclusivos para essa coluna são gerados automaticamente durante o processo de importação. Você não deve especificar um nome *id_column* que corresponda ao nome de qualquer coluna no arquivo dBASE. Algumas ferramentas espaciais requerem uma coluna com um identificador exclusivo.

Os requisitos e o efeito deste parâmetro dependem se a tabela já existe.

- Para uma tabela existente, o tipo de dados do parâmetro *id_column* pode ser qualquer tipo de número inteiro, como INTEGER, SMALLINT ou BIGINT.
- Para uma nova tabela, a coluna é definida da seguinte forma:

  ```
  INTEGER NOT NULL PRIMARY KEY
  ```

  Se *id_column_is_identity* tiver um valor diferente de zero, a definição será expandida da seguinte forma:

  ```
  INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY
  ( START WITH 1 INCREMENT BY 1 )
  ```

O valor de *id_column* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-idColumnIsIdentity *id_column_is_identity*
Indica se a *id_column* especificada será criada utilizando a cláusula IDENTITY. Se você especificar um valor diferente de zero em *id_column_is_identity*, a coluna *id_column* será criada como uma coluna de identidade. Este parâmetro será ignorado para tabelas que já existem.

-restartCount *restart_count*
Especifica que a operação de importação é iniciada com o registro $n + 1$. Os primeiros n registros serão ignorados. Se este parâmetro não for especificado, todos os registros, começando com o registro número 1, serão importados.

-commitScope *commit_scope*
Especifica que um COMMIT será executado após pelo menos menos n registros serem importados. Se este parâmetro não for especificado, será executado um COMMIT no final da operação. Isto pode resultar na utilização de um arquivo de log grande e na perda de dados em operações que são interrompidas.

-exceptionFile *exception_file*
Especifica o nome do caminho completo de um arquivo de forma no qual os dados de forma que não puderam ser importados serão gravados. Se o parâmetro não for especificado, não será criado um arquivo de exceção.

Se você especificar um valor para o parâmetro e incluir uma extensão do arquivo opcional, especifique .shp ou .SHP. Se você não especificar uma extensão, a extensão `.shp` será anexada a *exception_file*.

O arquivo de exceção contém o conjunto completo de linhas para a instrução de inserção que falhou. Uma instrução de inserção pode incluir diversas linhas. Por exemplo, suponha que uma linha não possa ser importada porque os dados de shape estão incorretamente codificados. Uma única instrução de inserção tenta importar 20 linhas, incluindo a linha com os dados de forma incorretos. Como a instrução de inserção falha, todo o conjunto de 20 linhas é gravado no arquivo de exceção.

Registros que são gravados no arquivo de exceção apenas quando esses registros podem ser identificados incorretamente, como é o caso quando o tipo de registro de forma não é válido. Alguns tipos de danos nos dados de shape

(arquivos .shp) e índice de shape (arquivos .shx) não permitem que os registros apropriados sejam identificados. Neste caso, os registros não podem ser gravados no arquivo de exceção e uma mensagem de erro é emitida para relatar o problema.

Se você especificar um valor para este parâmetro, quatro arquivos serão criados no servidor DB2. Consulte "Observações de Uso" para obter uma explicação destes arquivos.

Se o arquivo *exception_file* já existir, o comando retornará um erro.

O comprimento máximo para este parâmetro é de 256 caracteres.

-messagesFile *msg_file_name*
Especifica o nome do caminho completo do arquivo no servidor DB2 no qual o DB2 Spatial Extender grava mensagens sobre a operação de importação. Se você não especificar este parâmetro, o DB2 Spatial Extender não criará um arquivo de mensagens.

Os seguintes tipos de mensagens são gravados no arquivo de mensagens:

- Mensagens informativas como, por exemplo, um resumo da operação de importação
- Mensagens de erro para dados que não puderam ser importados, por exemplo, devido a diferentes sistemas de coordenadas. Estas mensagens de erro correspondem aos dados de forma que estão armazenados no arquivo de exceção *exception_file* especificado.

Se o arquivo *msg_file_name* já existir, o comando retornará um erro.

O comprimento máximo para este parâmetro é de 256 caracteres.

-client *client_flag*
Especifica se a operação de importação ocorre no cliente ou no servidor DB2 e onde os arquivos são criados. Os possíveis valores para este parâmetro são:

- 0 para indicar que a operação de importação ocorre no servidor DB2 e os arquivos são acessados a partir do servidor DB2.
- 1 para indicar que a operação de importação ocorre no cliente e os arquivos são acessados a partir do cliente.

Se você não especificar este parâmetro, o valor 0 será o padrão.

Observações de Uso

É possível executar o processo de importação no cliente no qual o comando é executado. Isto geralmente é mais conveniente já que não requer acesso ao sistema de arquivos do servidor DB2.

O **db2se import_shape** cria ou grava nos quatro arquivos a seguir:

- O arquivo shape principal (extensão .shp). Este arquivo é obrigatório.
- O arquivo shape de índice (extensão .shx). Este arquivo é opcional. Se ele existir, o desempenho da operação de importação poderá melhorar.
- Um arquivo dBASE que contém dados de atributo (extensão .dbf). Este arquivo é obrigatório apenas se os dados de atributo vão ser importados.
- O arquivo de projeção que especifica o sistema de coordenadas dos dados de shape (extensão .prj). Este arquivo é opcional. Se ele existir, o sistema de coordenadas definido nele será comparado com o sistema de coordenadas do sistema de referência espacial especificado pelo parâmetro *srs_id*.

A tabela a seguir descreve como os tipos de dados de atributo dBASE são mapeados para os tipos de dados DB2. Todos os outros tipos de dados de atributo não são suportados.

Tabela 21. Relacionamento entre os tipos de dados DB2 e os tipos de dados do atributo dBASE

Tipo .dbf	Comprimento de .dbf‰ (Consulte a nota)	Decimais de .dbf‰ (Consulte a nota)	Tipo SQL	Comentários
N	< 5	0	SMALLINT	
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 tabela inclui as variáveis a seguir, ambas são definidas no cabeçalho do arquivo dBASE:

- *len*, que representa o comprimento total da coluna no arquivo dBASE. O DB2 Spatial Extender utiliza este valor para duas finalidades:
  - Definir a precisão do tipo de dados SQL DECIMAL ou o comprimento do tipo de dados SQL CHAR
  - Determinar qual dos tipos de inteiro ou ponto flutuante devem ser utilizados
- *dec*, que representa o número máximo de dígitos à direita do ponto decimal da coluna no arquivo dBASE. O DB2 Spatial Extender utiliza este valor para definir a escala para o tipo de dados SQL DECIMAL.

Por exemplo, suponha que o arquivo dBASE contenha uma coluna de dados cujo comprimento (*len*) esteja definido como 20. Suponha que o número de dígitos à direita do ponto decimal (*dec*) seja definido como 5. Quando o DB2 Spatial Extender importa dados dessa coluna, ele utiliza os valores de *len* e *dec* para derivar o seguinte tipo de dados SQL: DECIMAL(20,5).

Exemplo

O comando a seguir importa os dados do arquivo de forma `myfile` localizado no cliente para a tabela MYTABLE. Os dados espaciais em `myfile` são inseridos na coluna MYCOLUMN na tabela MYTABLE.

```
db2se import_shape mydb -fileName myfile -srsName NAD83_SRS_1
        -tableName mytable -spatialColumnName mycolumn -client 1
```

Comando db2se register_gc

O comando **db2se register_gc** registra um geocodificador.

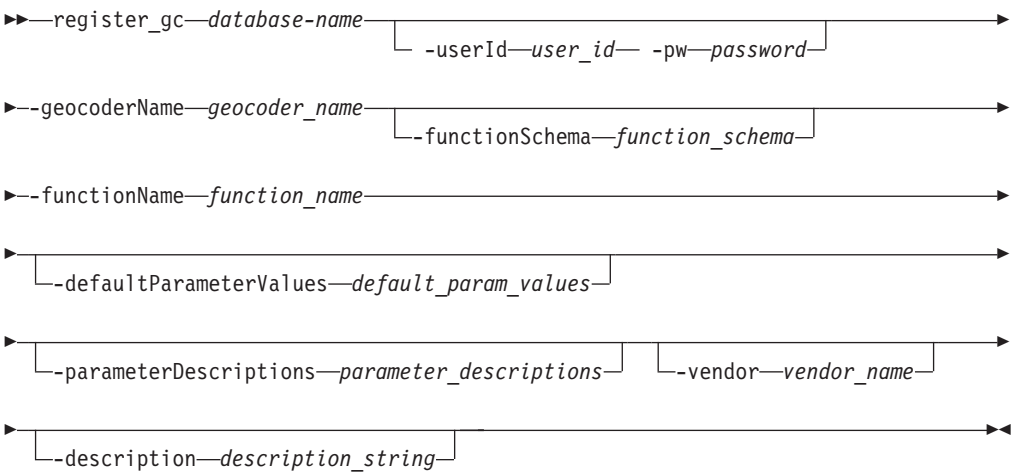
Este comando registra um geocodificador para que ele possa ser usado para geocodificar valores em uma tabela e armazenar ou atualizar o valor da geometria resultante. As informações sobre geocodificadores registrados estão disponíveis na visualização de catálogo DB2GSE.ST_GEOCODERS.

Autorização

O ID do usuário deve ter as autoridades DBADM e DATAACCESS no banco de dados espacial ativado para executar este comando.

Sintaxe do comando

Comando db2se register_gc



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja registrar um geocodificador.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-geocoderName *geocoder_name*
Identifica exclusivamente o geocodificador que você deseja registrar. O valor de *geocoder_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas. O comprimento máximo para este parâmetro é de 128 caracteres.

-functionSchema *function_schema*
Especifica o nome do esquema para a função que implementa este geocodificador. Se este parâmetro não for especificado, o valor no registro especial CURRENT SCHEMA será usado como o nome do esquema para a função.

O valor de *function_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-functionName *function_name*
Especifica o nome não qualificado da função que implementa este geocodificador. A função já deve estar criada e listada na visualização de catálogo SYSCAT.ROUTINES.

O valor *function_name*, juntamente com o valor *function_schema* definido implícita ou explicitamente, deve identificar exclusivamente a função.

O valor *function_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-defaultParameterValues *default_parameter_values*
Especifica a lista de valores de parâmetros padrão de codificação geográfica para a função do geocodificador.

Você deve especificar os valores de parâmetro na ordem em que foram definidos pela função e separá-los com uma vírgula. Por exemplo:

```
default_parm1_value,default_parm2_value,...
```

Cada valor de parâmetro deve ser uma expressão SQL. Siga estas diretrizes para especificar valores de parâmetro padrão:

- Se um valor for uma sequência, coloque-a entre aspas simples.
- Se um valor de parâmetro for um número, não coloque-o entre aspas simples.
- Se o valor do parâmetro for nulo, faça a coerção para o tipo correto. Por exemplo, especifique a seguinte expressão para indicar um NULL para um parâmetro de número inteiro:

  ```
  CAST(NULL AS INTEGER)
  ```
- Se o parâmetro de codificação geográfica tiver que ser uma coluna de codificação geográfica, não especifique um valor de parâmetro padrão.

Use duas vírgulas consecutivas (`...,,...`) para omitir valores para parâmetros que você indica ao configurar a codificação geográfica ou ao executá-la no modo em lote usando o parâmetro **-parameterValues** com o comando **db2se setup_gc** ou o comando **db2se run_gc**.

O comprimento máximo para este parâmetro é de 32.672 caracteres.

-parameterDescriptions *parameter_descriptions*
Especifica a lista de descrições de parâmetros de codificação geográfica para a função do geocodificador. O comprimento máximo para este parâmetro é de 32.672 caracteres.

Cada descrição de parâmetro que você especifica explica o significado e uso do parâmetro e pode conter até 256 caracteres. As descrições para os parâmetros devem ser separadas por vírgulas e devem aparecer na ordem dos parâmetros, conforme definido pela função. Para usar uma vírgula na descrição de um parâmetro, coloque a sequência entre aspas simples ou duplas. Por exemplo:

```
descrição,'descrição2, que contém uma vírgula',descrição3
```

-vendor *vendor_name*
Especifica o nome do fornecedor que implementou o geocodificador. O comprimento máximo para este parâmetro é de 128 caracteres.

-description *description_string*
Descreve o geocodificador, explicando sua finalidade. O comprimento máximo para este parâmetro é de 256 caracteres.

Observações de Uso

O tipo de retorno para a função do geocodificador deve corresponder ao tipo de dados da coluna geocodificada. Os parâmetros de codificação geográfica podem ser um nome da coluna (chamado de coluna de codificação geográfica) que contém dados de que o geocodificador precisa. Por exemplo, os parâmetros do

geocodificador podem identificar endereços ou um valor de significado específico para o geocodificador, tal como o score mínimo de correspondência. Se o parâmetro de codificação geográfica for um nome de coluna, a coluna deverá estar na mesma tabela ou exibição que a coluna geocodificada.

O tipo de retorno para a função do geocodificador serve como o tipo de dados para a coluna geocodificada. O tipo de retorno pode ser qualquer tipo de dados DB2, tipo definido pelo usuário ou tipo estruturado. Se um tipo definido pelo usuário ou estruturado for retornado, a função do geocodificador será responsável por retornar um valor válido do respectivo tipo de dados. Se a função do geocodificador retornar valores de um tipo espacial, ou seja, ST_Geometry ou um de seus subtipos, a função do geocodificador será responsável por construir uma geometria válida. A geometria deve ser representada utilizando um sistema de referência espacial existente. A geometria será válida se você chamar a função espacial ST_IsValid na geometria e um valor 1 for retornado. Os dados retornados da função do geocodificador são atualizados ou inseridos na coluna geocodificada, dependendo de qual operação (INSERT ou UPDATE) causou a geração do valor geocodificado.

Exemplo

O exemplo a seguir registra um geocodificador denominado “mygeocoder”, implementado por uma função denominada “myschema.myfunction”.

```
db2se register_gc mydb -geocoderName \"mygeocoder"\
        -functionSchema \"myschema\" -functionName \"myfnction\"
        -defaultParameterValues "1, 'string',,cast(null as varchar(50))"
        -vendor myvendor -description "myvendor geocoder
        returning well-known text"
```

Comando db2se register_spatial_column

O comando **db2se register_spatial_column** registra uma coluna espacial e associa a ela um sistema de referência espacial (SRS).

Registrar uma coluna espacial cria uma limitação na tabela, se possível, para assegurar que todas as geometrias utilizem o SRS especificado.

Além disso, é possível usar este comando para atualizar as informações de extensão espacial.

As informações sobre colunas espaciais registradas e extensões espaciais estão disponíveis na visualização de catálogo DB2GSE.ST_GEOMETRY_COLUMNS.

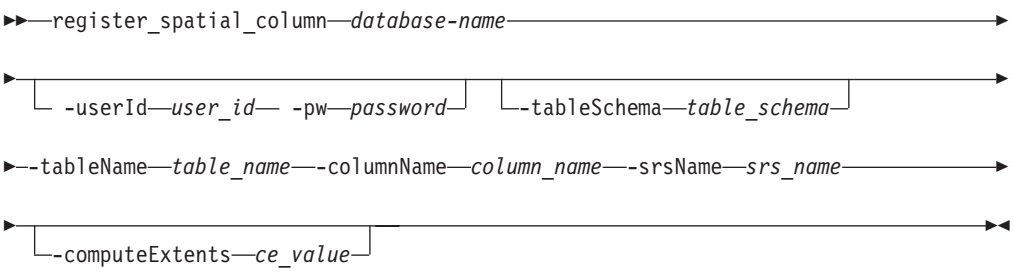
Autorização

O ID do usuário deve ter uma das seguintes autoridades ou privilégios para executar este comando:

- Autoridades DBADM e DATAACCESS no banco de dados que contém a tabela à qual pertence a coluna espacial que está sendo registrada
- Privilégio CONTROL nesta tabela
- Privilégio ALTER nesta tabela

Sintaxe do comando

Comando db2se register_spatial_column



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja registrar uma coluna espacial.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-tableSchema *table_schema*
Especifica o nome do esquema para o *table_name* especificado. Se você não especificar um nome do esquema, o valor no registro especial CURRENT SCHEMA será usado como o nome do esquema para a tabela ou visualização.

-tableName *table_name*
Especifica o nome não qualificado da tabela que contém a coluna que está sendo registrada. O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-columnName *column_name*
Identifica a coluna a ser registrada. O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-srsName *srs_name*
Identifica o sistema de referência espacial que deve ser usado para esta coluna espacial. O valor de *srs_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

computeExtents *ce_value*
Indica se serão calculadas as extensões geográficas de uma coluna especificada e se serão disponibilizadas por meio da visualização de catálogo DB2GSE.ST_GEOMETRY_COLUMNS. Os possíveis valores para este parâmetro são:

- Um valor maior que 0 para calcular as extensões geográficas.
- Um valor nulo, 0 ou negativo para impedir este cálculo.

Se você omitir este parâmetro, ele terá o mesmo efeito que especificar 0. O cálculo da extensão não é executado.

Exemplo

O exemplo a seguir registra uma coluna espacial denominada MYCOLUMN na tabela MYTABLE, com o sistema de referência espacial "USA_SRS_1".

```
db2se register_spatial_column mydb -tableName mytable -columnName mycolumn -srsName USA_SRS_1
```

Comando db2se remove_gc_setup

O comando **db2se remove_gc_setup** remove todas as informações de configuração de codificação geográfica para uma coluna geocodificada.

As informações associadas à coluna geocodificada especificada não estão mais disponíveis nas visualizações de catálogo DB2GSE.ST_GEOCODING e DB2GSE.ST_GEOCODING_PARAMETERS.

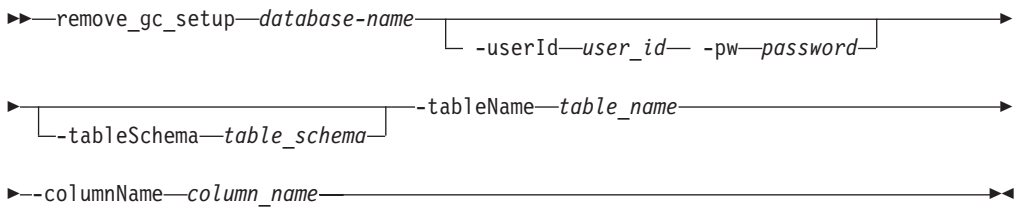
Autorização

O ID do usuário deve ter uma das seguintes autoridades ou privilégios para executar este comando:

- Autoridade DATAACCESS no banco de dados que contém a tabela na qual o geocodificador especificado irá operar
- Privilégio CONTROL nesta tabela
- Privilégio UPDATE nesta tabela

Sintaxe do comando

Comando db2se remove_gc_setup



Parâmetros de Comando

Em que:

database-name
: Especifica o nome do banco de dados para o qual você deseja remover todas as informações de configuração de codificação geográfica para uma coluna geocodificada.

-userId *user_id*
: Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
: Especifica a senha para *user_id*.

-tableSchema *table_schema*
: Especifica o nome do esquema para o *table_name* especificado. Se você não especificar um nome do esquema, o valor no registro especial CURRENT SCHEMA será usado como o nome do esquema para a tabela ou visualização.

-tableName *table_name*
Especifica o nome não qualificado da tabela para o *column_name* especificado. O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-columnName *column_name*
Identifica o nome da coluna da qual você deseja remover a configuração da codificação geográfica. O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

Observações de Uso

Não é possível remover uma configuração de geocodificação se geocodificação automática estiver ativada para a coluna geocodificada.

Exemplo

O exemplo a seguir remove uma configuração para operações de codificação geográfica que se aplicam a uma coluna espacial denominada MYCOLUMN na tabela MYTABLE.

```
db2se remove_gc_setup mydb -tableName mytable -columnName mycolumn
```

comando db2se restore_indexes

O comando **db2se restore_indexes** restaura os índices espaciais salvos anteriormente emitindo o comando **db2se save_indexes** para um banco de dados espacial ativado.

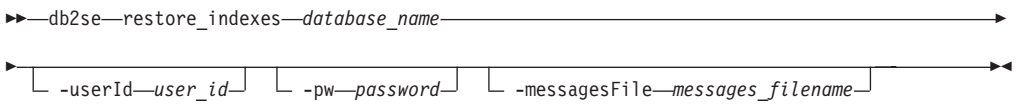
Este comando é usado para recriar índices espaciais após o upgrade de uma instância de 32 bits para 64 bits.

Autorização

Autoridades DBADM e DATAACCESS no banco de dados espacial ativado.

Sintaxe do comando

comando db2se restore_indexes



Parâmetros de Comando

database_name
O nome do banco de dados a ser feito o upgrade.

-userId user_id
O ID do usuário do banco de dados que possui autoridade SYSADM ou DBADM no banco de dados cujo upgrade está sendo feito.

-pw password
Sua senha de usuário.

-messagesFile messages_filename
O nome do arquivo que contém o relatório de ações de migração. O nome do arquivo fornecido deve ser um nome de arquivo completo no servidor.

comando db2se save_indexes

O comando db2se save_indexes salva os índices espaciais definidos em um banco de dados espacial ativado.

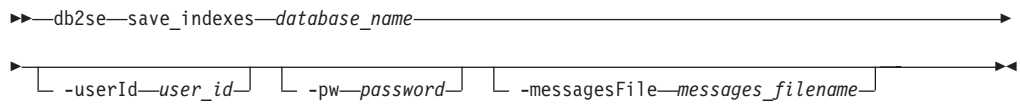
Este comando é usado para salvar definições de índice espacial atual ao fazer upgrade de uma instância de 32 bits para 64 bits.

Autorização

Autoridades DBADM e DATAACCESS no banco de dados espacial ativado.

Sintaxe do comando

comando db2se save_indexes



Parâmetros de Comando

database_name
O nome do banco de dados a ser feito o upgrade.

-userId user_id
O ID do usuário do banco de dados que possui autoridade SYSADM ou DBADM no banco de dados cujo upgrade está sendo feito.

-pw password
Sua senha de usuário.

-messagesFile messages_filename
O nome do arquivo que contém o relatório de ações de migração. O nome do arquivo fornecido deve ser um nome de arquivo completo no servidor.

Comando db2se run_gc

O comando **db2se run_gc** executa um geocodificador no modo em lote em uma coluna geocodificada.

As informações associadas à coluna geocodificada especificada não estão mais disponíveis nas visualizações de catálogo DB2GSE.ST_GEOCODING e DB2GSE.ST_GEOCODING_PARAMETERS.

Autorização

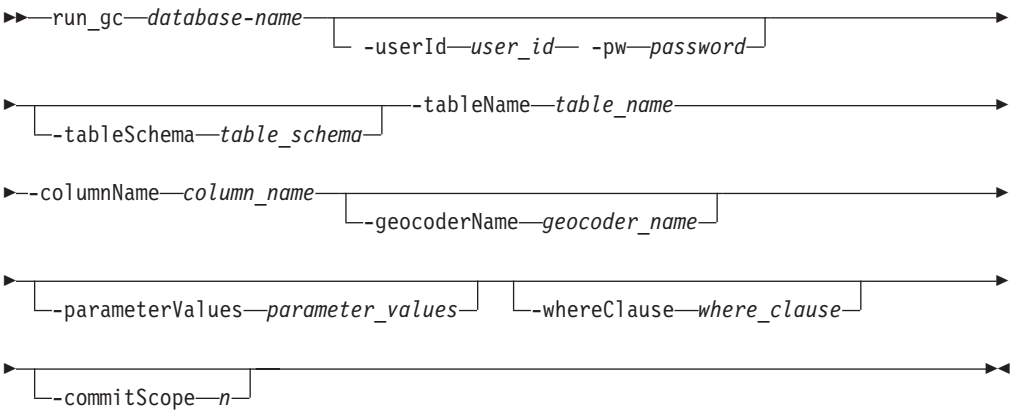
O ID do usuário deve ter uma das seguintes autoridades ou privilégios para executar este comando:

- Autoridade DATAACCESS no banco de dados que contém a tabela na qual o geocodificador especificado irá operar
- Privilégio CONTROL nesta tabela

- Privilégio UPDATE nesta tabela

Sintaxe do comando

Comando db2se run_gc



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja executar um geocodificador no modo em lote em uma coluna geocodificada.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-tableSchema *table_schema*
Especifica o nome do esquema para o *table_name* especificado. Se você não especificar um nome do esquema, o valor no registro especial CURRENT SCHEMA será usado como o nome do esquema para a tabela ou visualização.

-tableName *table_name*
Especifica o nome não qualificado da tabela para o *column_name* especificado. O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-columnName *column_name*
Identifica o nome da coluna na qual os dados geocodificados devem ser inseridos ou atualizados. O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-geocoderName *geocoder_name*
Identifica exclusivamente o geocodificador que deve executar a codificação geográfica. O valor de *geocoder_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas. O comprimento máximo para este parâmetro é de 128 caracteres.

-parameterValues *parameter_values*
Especifica a lista de valores de parâmetros de codificação geográfica para a função geocodificador. Se este parâmetro não for especificado, os valores

usados serão os valores de parâmetro que foram especificados durante a configuração do geocodificador ou os valores de parâmetro padrão que foram especificados durante o registro do geocodificador.

Você deve especificar os valores de parâmetro na ordem em que foram definidos pela função e separá-los com uma vírgula. Por exemplo:

```
default_parm1_value,default_parm2_value,...
```

Cada valor de parâmetro deve ser uma expressão SQL. Siga estas diretrizes para especificar valores de parâmetro padrão:

- Se um valor for uma sequência, coloque-a entre aspas simples.
- Se um valor de parâmetro for um número, não coloque-o entre aspas simples.
- Se o valor do parâmetro for nulo, faça a coerção para o tipo correto. Por exemplo, especifique a seguinte expressão para indicar um NULL para um parâmetro de número inteiro: `CAST(NULL AS INTEGER)`.
- Se o parâmetro de codificação geográfica tiver que ser uma coluna de codificação geográfica, não especifique um valor de parâmetro padrão.

Use duas vírgulas consecutivas (`...,,...`) para omitir valores para parâmetros para os quais você indicou um valor quando configurou ou registrou o geocodificador.

O comprimento máximo para este parâmetro é de 32.672 caracteres.

-whereClause *where_clause*
Especifica o texto para uma condição de procura de uma cláusula WHERE para filtrar o conjunto de registros a serem geocodificados. Se este parâmetro não for especificado, o valor *where_clause* especificado durante a configuração da codificação geográfica será usado. Se um valor *where_clause* não foi especificado durante a configuração da codificação geográfica, todas as linhas na tabela serão geocodificadas.

Você pode especificar uma cláusula que faz referência a qualquer coluna na tabela ou exibição em que o geocodificador irá operar.

Não especifique a palavra-chave WHERE em *where_clause*.

O comprimento máximo para este parâmetro é de 32.672 caracteres.

-commitScope *n*
Especifica que um COMMIT deve ser executado após a codificação geográfica de cada *n* registros. Se este parâmetro não for especificado, o valor especificado com o parâmetro **-commitScope** durante a configuração da codificação geográfica será usado. Se este parâmetro não for especificado e um valor não tiver sido especificado durante a configuração da codificação geográfica, será executado um COMMIT no final da operação. Um COMMIT no final da operação pode resultar na utilização de um arquivo de log grande e na perda de dados em operações que são interrompidas.

Exemplo

O exemplo a seguir executa um geocodificador no modo em lote para ocupar uma coluna denominada MYCOLUMN em uma tabela denominada MYTABLE.

```
db2se run_gc mydb -tableName mytable -columnName mycolumn
```

Comando db2se setup_gc

O comando **db2se setup_gc** associa uma coluna que deve ser geocodificada com um geocodificador e para configurar os parâmetros de codificação geográfica correspondentes.

As informações sobre esta configuração estão disponíveis nas visualizações de catálogo DB2GSE.ST_GEOCODING e DB2GSE.ST_GEOCODING_PARAMETERS.

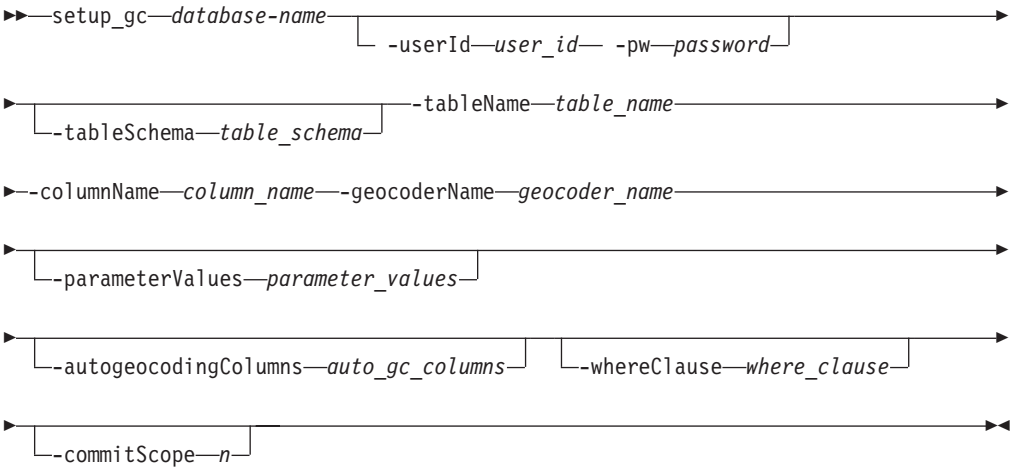
Autorização

O ID do usuário deve ter uma das seguintes autoridades ou privilégios para executar este comando:

- Autoridade DATAACCESS no banco de dados que contém a tabela na qual o geocodificador especificado irá operar
- Privilégio CONTROL nesta tabela
- Privilégio UPDATE nesta tabela

Sintaxe do comando

Comando db2se setup_gc



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja configurar um geocodificador.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-tableSchema *table_schema*
Especifica o nome do esquema para o *table_name* especificado. Se você não especificar um nome do esquema, o valor no registro especial CURRENT SCHEMA será usado como o nome do esquema para a tabela ou visualização.

-tableName *table_name*
Especifica o nome não qualificado da tabela para o *column_name* especificado. O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-columnName *column_name*
Identifica o nome da coluna na qual os dados geocodificados devem ser inseridos ou atualizados. O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-geocoderName *geocoder_name*
Identifica exclusivamente um geocodificador registrado anteriormente que deve executar a codificação geográfica. O valor de *geocoder_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas. O comprimento máximo para este parâmetro é de 128 caracteres.

-parameterValues *parameter_values*
Especifica a lista de valores de parâmetros de codificação geográfica para a função geocodificador. Se este parâmetro não for especificado, serão utilizados os valores de parâmetros padrão especificados ao registrar o geocodificador.

Você deve especificar os valores de parâmetro na ordem em que foram definidos pela função e separá-los com uma vírgula. Por exemplo:

```
default_parm1_value,default_parm2_value,...
```

Cada valor de parâmetro deve ser uma expressão SQL. Siga estas diretrizes para especificar valores de parâmetro padrão:

- Se um valor for uma sequência, coloque-a entre aspas simples.
- Se um valor de parâmetro for um número, não coloque-o entre aspas simples.
- Se o valor do parâmetro for nulo, faça a coerção para o tipo correto. Por exemplo, especifique a seguinte expressão para indicar um NULL para um parâmetro de número inteiro: `CAST(NULL AS INTEGER)`.
- Se o parâmetro de codificação geográfica tiver que ser uma coluna de codificação geográfica, não especifique um valor de parâmetro padrão.

Use duas vírgulas consecutivas (`...,,...`) para omitir os valores para os parâmetros indicados ao configurar ou registrar o geocodificador.

O comprimento máximo para este parâmetro é de 32.672 caracteres.

-whereClause *where_clause*
Especifica o texto para uma condição de procura de uma cláusula WHERE para filtrar o conjunto de registros a serem geocodificados. Se este parâmetro não for especificado, todas as linhas na tabela serão geocodificadas.

Você pode especificar uma cláusula que faz referência a qualquer coluna na tabela ou exibição em que o geocodificador irá operar.

Não especifique a palavra-chave WHERE em *where_clause*.

O comprimento máximo para este parâmetro é de 32.672 caracteres.

-commitScope *n*
Especifica que um COMMIT deve ser executado após a codificação geográfica de cada *n* registros. Se este parâmetro não for especificado, será executado um COMMIT no final da operação. Um COMMIT no final da operação pode resultar na utilização de um arquivo de log grande e na perda de dados em operações que são interrompidas.

Observações de Uso

Este comando não chama a codificação geográfica. Ele fornece um modo para especificar as definições de parâmetros para a coluna que será geocodificada. As configurações de parâmetro que são especificadas na configuração de codificação geográfica substituem qualquer um dos valores de parâmetro padrão especificados no registro do geocodificador.

Você deve executar este comando antes de ativar a geocodificação automática. A configuração dos parâmetros de codificação geográfica é necessária antes da ativação da geocodificação automática.

É possível executar este comando antes da codificação geográfica no modo em lote. Se você não especificar nenhum valor de parâmetro para executar a codificação geográfica no modo em lote, os valores de parâmetro especificados na configuração de codificação geográfica serão usados. Se você especificar valores de parâmetro, estes valores substituirão os valores de parâmetro especificados na configuração de codificação geográfica.

Exemplo

O exemplo a seguir configura operações de codificação geográfica para ocupar uma coluna espacial denominada MYCOLUMN na tabela MYTABLE.

```
db2se setup_gc mydb -tableName mytable -columnName mycolumn
        -parameterValues "address,city,state,zip,2,90,70,20,1.1,'meter',4.."
        -autogeocodingColumns address,city,state,zip
        -commitScope 10
```

Comando db2se shape_info

O comando **db2se shape_info** mostra informações sobre um arquivo de forma e seu conteúdo. Para um banco de dados especificado, opcionalmente, este comando pode exibir todos os sistemas de coordenadas compatíveis e sistemas de referência espacial que estão incluídos no arquivo de forma.

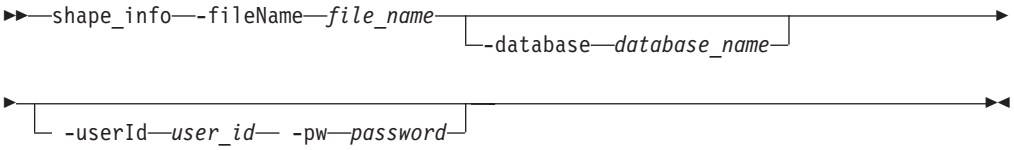
Autorização

O ID do usuário deve ter os privilégios necessários no servidor DB2 para ler os arquivos de forma.

Se o parâmetro **-database** for especificado, o ID do usuário deverá ter o privilégio SELECT na visualização de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Sintaxe do comando

Comando db2se shape_info



Parâmetros de Comando

Em que:

-fileName *file_name*
Especifica o nome do caminho completo do arquivo de forma sobre o qual você deseja exibir informações.

O DB2 Spatial Extender primeiro procura uma correspondência exata do nome especificado com o parâmetro **-fileName**. Se o DB2 Spatial Extender não localizar uma correspondência exata, ele procurará primeiro um arquivo com a extensão `.shp` e, em seguida, um arquivo com a extensão `.SHP`.

O comprimento máximo para este parâmetro é de 256 caracteres.

-database *database_name*
Especifica o nome do banco de dados para o qual você deseja localizar todos os sistemas de coordenadas e sistemas de referência espacial compatíveis no arquivo de forma *file_name*.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

Exemplo

O exemplo a seguir mostra informações sobre um arquivo de forma chamado `MYFILE`, que está localizado no diretório atual.

```
db2se shape_info -fileName myfile
```

O exemplo a seguir mostra informações sobre um arquivo shape de amostra do UNIX denominado `offices`. O parâmetro *–database* localiza todos os sistemas de coordenadas e de referência espacial compatíveis no banco de dados indicado (neste caso, MYDB).

```
db2se shape_info -fileName ~/sqllib/samples/extenders/spatial/data/offices
                 -database myDB
```

Comando db2se unregister_gc

O comando **db2se unregister_gc** cancela o registro de um geocodificador.

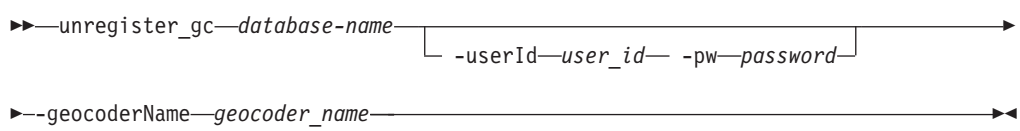
Para procurar informações sobre o geocodificador que você deseja cancelar o registro, consulte a exibição do catálogo DB2GSE.ST_GEOCODERS.

Autorização

O ID do usuário deve ter as autoridades DBADM e DATAACCESS no banco de dados espacial ativado para executar este comando.

Sintaxe do comando

Comando db2se unregister_gc



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja cancelar o registro de um geocodificador.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-geocoderName *geocoder_name*
Identifica exclusivamente o geocodificador do qual você deseja cancelar o registro. O valor de *geocoder_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas. O comprimento máximo para este parâmetro é de 128 caracteres.

Observações de Uso

Não é possível cancelar o registro de um geocodificador se ele for especificado na configuração de geocodificação de alguma coluna. Para determinar se um geocodificador está especificado na configuração de codificação geográfica de uma coluna, verifique as exibições do catálogo DB2GSE.ST_GEOCODING e DB2GSE.ST_GEOCODING_PARAMETERS.

Exemplo

O exemplo a seguir cancela o registro de um geocodificador chamado MYGEOCODER.

```
db2se unregister_gc mydb -geocoderName mygeocoder
```

Comando db2se unregister_spatial_column

O comando **db2se unregister_spatial_column** remove o registro de uma coluna espacial.

Este comando remove o registro:

- Removendo a associação do sistema de referência espacial com a coluna espacial. A visualização de catálogo DB2GSE.ST_GEOMETRY_COLUMNS continua mostrando a coluna espacial, mas a coluna não está mais associada a nenhum sistema de referência espacial.
- Para uma tabela base, elimine a restrição criada nesta tabela para assegurar que os valores de geometria nesta coluna espacial sejam todos representados no mesmo sistema de referência espacial.

Autorização

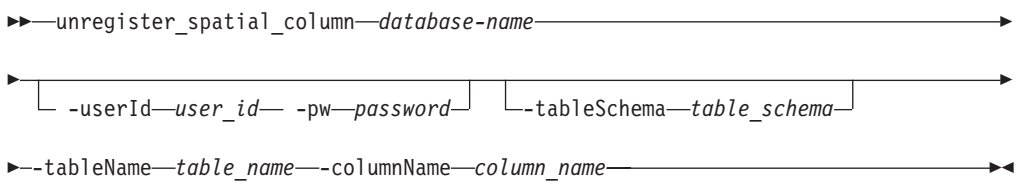
O ID do usuário deve ter uma das seguintes autoridades ou privilégios para executar este comando:

- Autoridades DBADM e DATAACCESS no banco de dados que contém a tabela à qual pertence a coluna espacial que está sendo registrada
- Privilégio CONTROL nesta tabela

- Privilégio ALTER nesta tabela

Sintaxe do comando

Comando db2se unregister_spatial_column



Parâmetros de Comando

Em que:

database-name
Especifica o nome do banco de dados para o qual você deseja remover o registro de uma coluna espacial.

-userId *user_id*
Especifica o ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados indicado por *database_name*.

-pw *password*
Especifica a senha para *user_id*.

-tableSchema *table_schema*
Especifica o nome do esquema para o *table_name* especificado. Se você não especificar um nome do esquema, o valor no registro especial CURRENT SCHEMA será usado como o nome do esquema para a tabela ou visualização.

-tableName *table_name*
Especifica o nome não qualificado da tabela para o *column_name* especificado. O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

-columnName *column_name*
Identifica a coluna da qual um registro será cancelado. O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

Exemplo

O exemplo a seguir cancela registros de uma coluna espacial denominada MYCOLUMN na tabela MYTABLE.

```
db2se unregister_spatial_column mydb -tableName mytable -columnName mycolumn
```

Comando db2se upgrade

O comando **db2se upgrade** faz upgrade de um banco de dados espacial ativado da Versão 9.5 ou da Versão 9.7 para o Versão 10.1.

Este comando pode descartar e recriar índices espaciais para concluir o upgrade do banco de dados, que pode utilizar uma quantidade significativa de tempo, dependendo dos tamanhos das tabelas. Por exemplo, seus índices serão eliminados e recriados se seus dados estiverem sendo movidos de uma instância de 32 bits para uma instância de 64 bits.

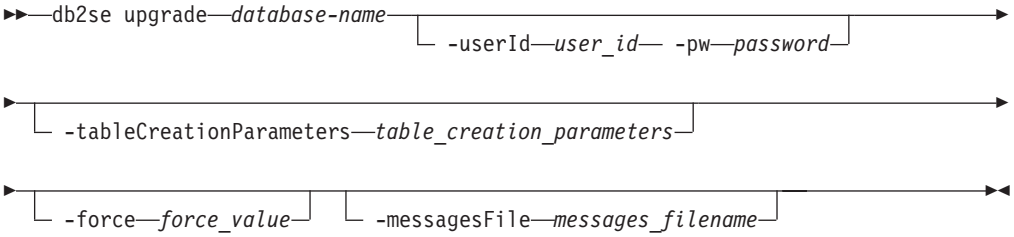
Dica: Execute o comando **db2se upgrade** com a opção `-force 0` e especifique um arquivo de mensagens para descobrir de quais índices o upgrade deve ser feito sem executar o processamento de upgrade adicional.

Autorização

Autoridades DBADM e DATAACCESS no banco de dados espacial ativado do qual você deseja fazer upgrade.

Sintaxe do comando

Comando db2se upgrade



Parâmetros de Comando

Em que:

database_name
O nome do banco de dados a ser feito o upgrade.

-userId user_id
O ID do usuário do banco de dados que possui autoridade DATAACCESS no banco de dados do qual está sendo feito o upgrade.

-pw password
Sua senha de usuário.

-tableCreationParameters table_creation_parameters
Os parâmetros a serem utilizados na criação das tabelas do catálogo Spatial Extender.

-force force_value

- 0: Valor padrão. Tentará a migração do banco de dados, mas pára se quaisquer objetos definidos pelo aplicativo, como visualizações, funções, acionadores ou índices espaciais foram baseados em objetos Spatial Extender.
- 1: Salva e restaura automaticamente objetos definidos pelo aplicativo. Salva e restaura índices espaciais, se necessário.
- 2: Salva e restaura automaticamente objetos definidos pelo aplicativo. Salva informações sobre o índice espacial, mas não restaura índices espaciais automaticamente.

-messagesFile messages_filename
O nome do arquivo que contém o relatório das ações de upgrade. O nome do arquivo fornecido deve ser um nome de arquivo completo no servidor.

Dica: Especifique este parâmetro para ajudar a resolver problemas de upgrade.

Restrição: Não é possível especificar um arquivo existente.

Observações de Uso

O comando **db2se upgrade** verifica várias condições e retorna um ou mais dos seguintes erros se qualquer uma dessas condições não for verdadeira:

- O banco de dados não está ativado espacialmente.
- O nome do banco de dados não é válido.
- Existem outras conexões com o banco de dados. Não é possível continuar com o upgrade.
- O catálogo espacial não está consistente.
- O usuário não está autorizado.
- A senha não é válida.
- Não foi possível fazer upgrade de alguns objetos definidos pelo usuário.

Certifique-se de que você tenha um espaço de tabela temporário do sistema com um tamanho de página de 8 KB ou maior e com um tamanho mínimo de 500 páginas. Isso é um requisito para executar o comando **db2se upgrade** com êxito.

comando db2se migrate

O comando **db2se migrate** migra um banco de dados ativado espacialmente para a Versão 9.7. Esse comando foi reprovado e será removido em um release futuro. Em vez disso, use o comando **db2se upgrade**.

Este comando pode eliminar e recriar índices espaciais para concluir a migração, que pode utilizar uma quantidade significativa de tempo dependendo dos tamanhos de suas tabelas. Por exemplo, seus índices serão eliminados e recriados se seus dados estiverem sendo movidos de uma instância de 32 bits para uma instância de 64 bits.

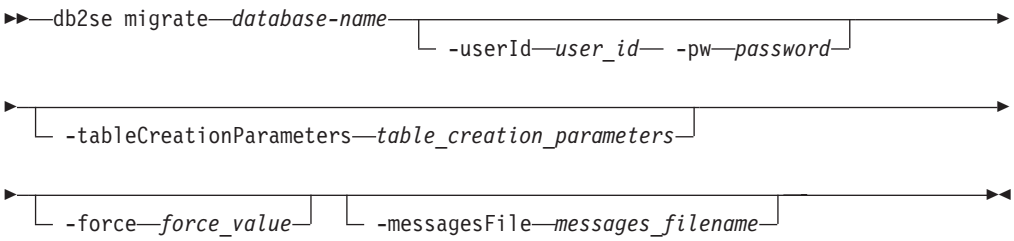
Dica: Execute o comando **db2se migrate** com a opção `-force 0` e especifique um arquivo de mensagens para descobrir quais índices devem ser migrados sem desempenhar processamento de migração adicional.

Autorização

Autoridade SYSADM ou DBADM no banco de dados espacialmente ativado que deseja migrar.

Sintaxe do comando

comando db2se migrate



Parâmetros de Comando

Em que:

database_name
O nome do banco de dados a ser migrado.

-userId user_id
O ID do usuário do banco de dados que tem autoridade SYSADM ou DBADM no banco de dados que está sendo migrado.

-pw password
Sua senha de usuário.

-tableCreationParameters table_creation_parameters
Os parâmetros a serem utilizados na criação das tabelas do catálogo Spatial Extender.

-force force_value

- 0: Valor padrão. Tentará a migração do banco de dados, mas pára se quaisquer objetos definidos pelo aplicativo, como visualizações, funções, acionadores ou índices espaciais foram baseados em objetos Spatial Extender.
- 1: Salva e restaura automaticamente objetos definidos pelo aplicativo. Salva e restaura índices espaciais, se necessário.
- 2: Salva e restaura automaticamente objetos definidos pelo aplicativo. Salva informações sobre o índice espacial, mas não restaura índices espaciais automaticamente.

-messagesFile messages_filename
O nome do arquivo que contém o relatório de ações de migração. O nome do arquivo fornecido deve ser um nome de arquivo completo no servidor.

Dica: Especifique este parâmetro para ajudar a resolver problemas de migração.

Restrição: Não é possível especificar um arquivo existente.

Você pode receber um ou mais dos seguintes erros durante a migração:

- O banco de dados não está ativado espacialmente.
- O nome do banco de dados não é válido.
- Existem outras conexões com o banco de dados. Executar agora...
- O catálogo espacial não está consistente.
- O usuário não está autorizado.
- A senha não é válida.
- Alguns objetos do usuário não puderam ser migrados.

Capítulo 17. Procedimentos armazenados

Use os procedimentos armazenados do DB2 Spatial Extender para configurar o DB2 Spatial Extender e criar projetos que usam dados espaciais.

Ao configurar o DB2 Spatial Extender ou o processador de linha de comandos do DB2, você chama esses procedimentos armazenados implicitamente. Por exemplo, quando você emite o comando de CLP **db2se create_srs**, o procedimento DB2GSE.ST_CREATE_SRS é chamado.

Alternativamente, é possível chamar os procedimentos armazenados do DB2 Spatial Extender explicitamente em um programa de aplicativo.

Antes de chamar a maioria dos procedimentos armazenados do DB2 Spatial Extender em um banco de dados, execute as tarefas a seguir:

1. Certifique-se de que você tenha um espaço de tabela temporário do sistema com um tamanho de página de 8 KB ou maior e com um tamanho mínimo de 500 páginas. Este é um requisito para poder executar com êxito o procedimento ST_ENABLE_DB STORED ou o comando **db2se enable_db**.
2. Ative o banco de dados para operações espaciais chamando o procedimento ST_ENABLE_DB. Consulte "Procedimento ST_ENABLE_DB" na página 196 para obter detalhes.

Depois que um banco de dados é ativado para operações espaciais, você pode chamar qualquer procedimento armazenado do DB2 Spatial Extender, implicitamente ou explicitamente, nesse banco de dados, se estiver conectado ao mesmo.

Este capítulo fornece tópicos para todos os procedimentos armazenados do DB2 Spatial Extender, conforme a seguir:

- "Procedimento ST_UNREGISTER_SPATIAL_COLUMN" na página 225

As implementações dos procedimentos armazenados estão arquivadas na biblioteca db2gse do servidor do DB2 Spatial Extender.

Procedimento ST_ALTER_COORDSYS

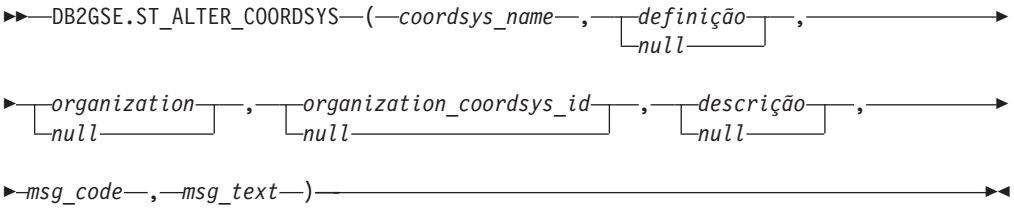
Utilize este procedimento armazenado para atualizar uma definição do sistema de coordenadas no banco de dados. Quando este procedimento armazenado é processado, as informações sobre o sistema de coordenadas são atualizadas na exibição do catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Atenção: Utilize este procedimento armazenado com atenção. Se você utilizar este procedimento para alterar a definição do sistema de coordenadas, e tiver dados espaciais existentes que estejam associados a um sistema de referência espacial que baseia-se nesse sistema de coordenadas, os dados espaciais poderão ser alterados inadvertidamente. Se os dados espaciais forem afetados, você será responsável por assegurar que os dados espaciais alterados ainda sejam precisos e válidos.

Autorização

O ID do usuário, sob o qual o procedimento armazenado é invocado, deve ter autoridade DBADM.

Sintaxe



Descrições de parâmetros

coordsys_name
Identifica exclusivamente o sistema de coordenadas. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *coordsys_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

definition
Defina o sistema de coordenadas. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, a definição do sistema de coordenadas não será alterado.

O tipo de dados deste parâmetro é VARCHAR(2048).

organization
Nomeia a organização que definiu o sistema de coordenadas e forneceu a

definição do mesmo; por exemplo, "European Petroleum Survey Group (EPSG)". Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo.

Se este parâmetro for nulo, a organização do sistema de coordenadas não foi alterado. Se este parâmetro não for nulo, o parâmetro *organization_coordsys_id* não poderá ser nulo; nesse caso, a combinação dos parâmetros *organization* e *organization_coordsys_id* identifica exclusivamente o sistema de coordenadas.

O tipo de dados deste parâmetro é VARCHAR(128).

organization_coordsys_id
Especifica um identificador numérico que está associado a este sistema de coordenadas pela entidade listada no parâmetro *organization*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo.

Se este parâmetro for nulo, o parâmetro *organization* também deverá ser nulo; nesse caso, o identificador do sistema de coordenadas da organização não é alterado. Se este parâmetro não for nulo, o parâmetro *organization* não poderá ser nulo; nesse caso, a combinação dos parâmetros *organization* e *organization_coordsys_id* identifica exclusivamente o sistema de coordenadas.

O tipo de dados deste parâmetro é INTEGER.

descrição
Descreve o sistema de coordenadas, explicando sua finalidade. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, as informações de descrição sobre o sistema de coordenadas não serão alteradas.

O tipo de dados deste parâmetro é VARCHAR(256).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_ALTER_COORDSYS. Este exemplo utiliza um comando DB2 CALL para atualizar um sistema de coordenadas denominado NORTH_AMERICAN_TEST. Este comando CALL atribui um valor de 1002 ao parâmetro *coordsys_id*:

```
call DB2GSE.ST_ALTER_COORDSYS('NORTH_AMERICAN_TEST',NULL,NULL,1002,NULL,?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_ALTER_SRS

Utilize este procedimento armazenado para atualizar uma definição do sistema de referência espacial no banco de dados. Quando este procedimento armazenado é processado, as informações sobre o sistema de referência espacial são atualizadas na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Internamente, o DB2 Spatial Extender armazena os valores de coordenada como inteiros positivos. Portanto, durante o cálculo, o impacto de erros de arredondamento (que são seriamente dependentes do valor real para operações de ponto flutuante) pode ser reduzido. O desempenho das operações espaciais também pode melhorar significativamente.

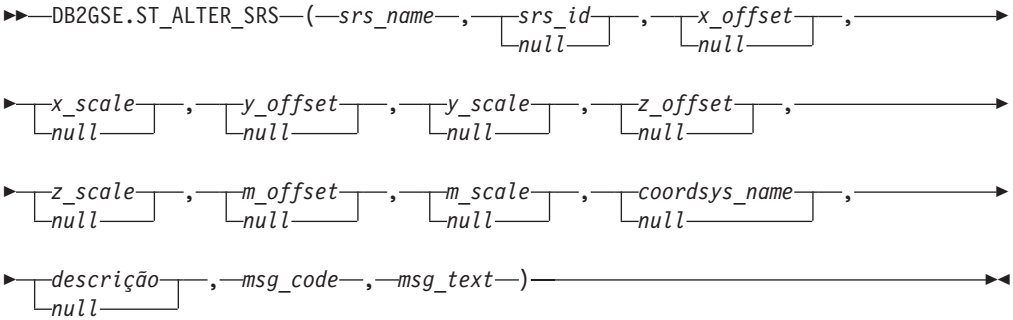
Restrição: Você não pode alterar um sistema de referência espacial se uma coluna espacial registrada o utiliza.

Atenção: Utilize este procedimento armazenado com atenção. Se você utilizar este procedimento para alterar o deslocamento, escala ou parâmetros *coordsys_name* do sistema de referência espacial, e tiver dados espaciais associados ao sistema de referência espacial, os dados espaciais poderão ser alterados inadvertidamente. Se os dados espaciais forem afetados, você será responsável por assegurar que os dados espaciais alterados ainda sejam precisos e válidos.

Autorização

O ID do usuário, sob o qual o procedimento armazenado é invocado, deve ter autoridade DBADM.

Sintaxe



Descrições de parâmetros

srs_name
Identifica o sistema de referência espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *srs_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

srs_id
Identifica exclusivamente o sistema de referência espacial. Este identificador é utilizado como um parâmetro de entrada para várias funções espaciais. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o identificador numérico do sistema de referência espacial não será alterado.

O tipo de dados deste parâmetro é INTEGER.

x_offset
Especifica o deslocamento de todas as coordenadas X de geometrias que são representadas neste sistema de referência espacial. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor para este parâmetro na definição do sistema de referência espacial não será alterado.

O deslocamento é subtraído antes do fator de escala *x_scale* ser aplicado quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender. (*WKT* é texto reconhecido e *WKB* é binário reconhecido).

O tipo de dados deste parâmetro é DOUBLE.

x_scale
Especifica o fator de escala de todas as coordenadas X de geometrias que são representadas neste sistema de referência espacial. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor para este parâmetro na definição do sistema de referência espacial não será alterado.

O fator de escala é aplicado (multiplicação) após o deslocamento *x_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender.

O tipo de dados deste parâmetro é DOUBLE.

y_offset
Especifica o deslocamento de todas as coordenadas Y de geometrias que são representadas neste sistema de referência espacial. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor para este parâmetro na definição do sistema de referência espacial não será alterado.

O deslocamento é subtraído antes do fator de escala *y_scale* ser aplicado quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender.

O tipo de dados deste parâmetro é DOUBLE.

y_scale
Especifica o fator de escala de todas as coordenadas Y de geometrias que são representadas neste sistema de referência espacial. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor para este parâmetro na definição do sistema de referência espacial não será alterado.

O fator de escala é aplicado (multiplicação) após o deslocamento *y_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender Este fator de escala deve ser igual a *x_scale*.

O tipo de dados deste parâmetro é DOUBLE.

z_offset
Especifica o deslocamento de todas as coordenadas Z de geometrias que são representadas neste sistema de referência espacial. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor para este parâmetro na definição do sistema de referência espacial não será alterado.

O deslocamento é subtraído antes do fator de escala *z_scale* ser aplicado quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender.

O tipo de dados deste parâmetro é DOUBLE.

z_scale
Especifica o fator de escala de todas as coordenadas Z de geometrias que são representadas neste sistema de referência espacial. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor para este parâmetro na definição do sistema de referência espacial não será alterado.

O fator de escala é aplicado (multiplicação) após o deslocamento *z_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender.

O tipo de dados deste parâmetro é DOUBLE.

m_offset
Especifica o deslocamento de todas as coordenadas M de geometrias que são representadas neste sistema de referência espacial. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor para este parâmetro na definição do sistema de referência espacial não será alterado.

O deslocamento é subtraído antes do fator de escala *m_scale* ser aplicado quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender.

O tipo de dados deste parâmetro é DOUBLE.

m_scale
Especifica o fator de escala de todas as coordenadas M de geometrias que são representadas neste sistema de referência espacial. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor para este parâmetro na definição do sistema de referência espacial não será alterado.

O fator de escala é aplicado (multiplicação) após o deslocamento *m_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender.

O tipo de dados deste parâmetro é DOUBLE.

coordsys_name
Identifica exclusivamente o sistema de coordenadas no qual se baseia este sistema de referência espacial. O sistema de coordenadas deve estar listado na exibição ST_COORDINATE_SYSTEMS. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o sistema de coordenadas que é utilizado para este sistema de referência espacial não será alterado.

O valor de *coordsys_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

descrição
Descreve o sistema de referência espacial, explicando sua finalidade. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, as informações de descrição sobre o sistema de referência espacial não será alterado.

O tipo de dados deste parâmetro é VARCHAR(256).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_ALTER_SRS. Este exemplo utiliza um comando DB2 CALL para alterar o valor de parâmetro *description* de um sistema de referência 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',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

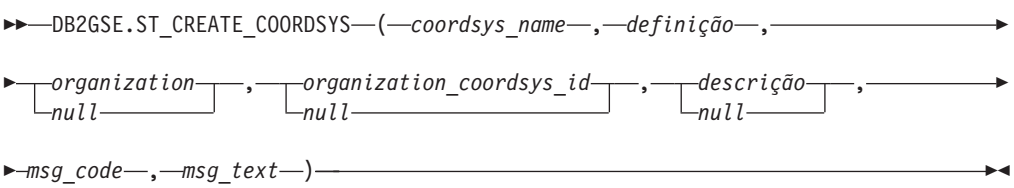
Procedimento ST_CREATE_COORDSYS

Use este procedimento armazenado para criar um novo sistema de coordenadas. Quando este procedimento armazenado é processado, as informações sobre o sistema de coordenadas são incluídas na exibição do catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Autorização

O ID do usuário, sob o qual o procedimento armazenado é invocado, deve ter autoridade DBADM.

Sintaxe



Descrições de parâmetros

coordsys_name
Identifica exclusivamente o sistema de coordenadas. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *coordsys_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

definition
Defina o sistema de coordenadas. Você deve especificar um valor diferente de nulo para este parâmetro. O fornecedor do sistema de coordenadas geralmente possui as informações para este parâmetro.

O tipo de dados deste parâmetro é VARCHAR(2048).

organization
Nomeia a organização que definiu o sistema de coordenadas e forneceu a definição do mesmo; por exemplo, "European Petroleum Survey Group (EPSG)". Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo.

Se este parâmetro for nulo, o parâmetro *organization_coordsys_id* também deverá ser nulo. Se este parâmetro não for nulo, o parâmetro *organization_coordsys_id* não poderá ser nulo; nesse caso, a combinação dos parâmetros *organization* e *organization_coordsys_id* identifica exclusivamente o sistema de coordenadas.

O tipo de dados deste parâmetro é VARCHAR(128).

organization_coordsys_id
Especifica um identificador numérico. A entidade que é especificada no parâmetro *organization* atribui este valor. O valor não é necessariamente exclusivo entre todos sistemas de coordenadas. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo.

Se este parâmetro for nulo, o parâmetro *organization* também deverá ser nulo. Se este parâmetro não for nulo, o parâmetro *organization* não poderá ser nulo; nesse caso, a combinação dos parâmetros *organization* e *organization_coordsys_id* identifica exclusivamente o sistema de coordenadas.

O tipo de dados deste parâmetro é INTEGER.

descrição
Descreve o sistema de coordenadas, explicando sua finalidade. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, nenhuma informação de descrição sobre o sistema de coordenadas será gravada.

O tipo de dados deste parâmetro é VARCHAR(256).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_CREATE_COORDSYS. Este exemplo utiliza um comando DB2 CALL para criar um sistema de coordenadas com os seguintes valores de parâmetros:

- Parâmetro *coordsys_name*: NORTH_AMERICAN_TEST
- Parâmetro *definition*:

```
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]]
```

- Parâmetro *organization*: EPSG
- Parâmetro *organization_coordsys_id*: 1001
- Parâmetro *description*: Teste de Sistemas de Coordenadas

```
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,'Teste de Sistemas de Coordenadas',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_CREATE_SRS

Use o procedimento armazenado ST_CREATE_SRS para criar um sistema de referência espacial.

Um sistema de referência espacial é definido pelo sistema de coordenadas, pela precisão e pelas extensões de coordenadas que são representadas no sistema de referência espacial. As extensões são os valores de coordenadas mínimo e máximo possíveis para as coordenadas X, Y, Z e M.

Internamente, o DB2 Spatial Extender armazena os valores de coordenada como inteiros positivos. Portanto, durante o cálculo, o impacto de erros de

arredondamento (que são seriamente dependentes do valor real para operações de ponto flutuante) pode ser reduzido. O desempenho das operações espaciais também pode melhorar significativamente.

Este procedimento armazenado possui duas variações:

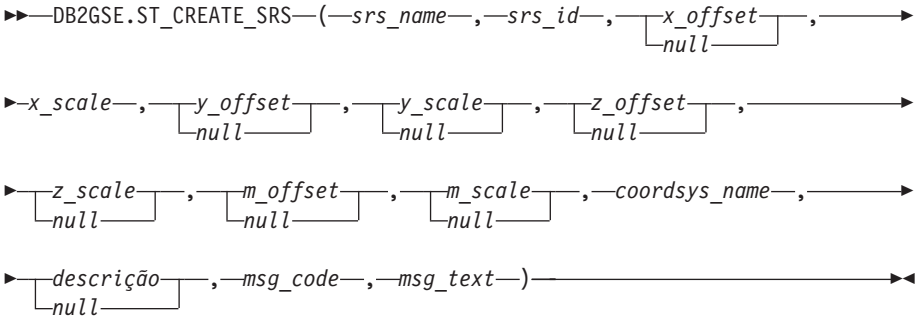
- A primeira variação utiliza os fatores de conversão (deslocamentos e fatores de escala) como parâmetros de entrada.
- A segunda variação utiliza as extensões e a precisão como parâmetros de entrada e calcula os fatores de conversão internamente.

Autorização

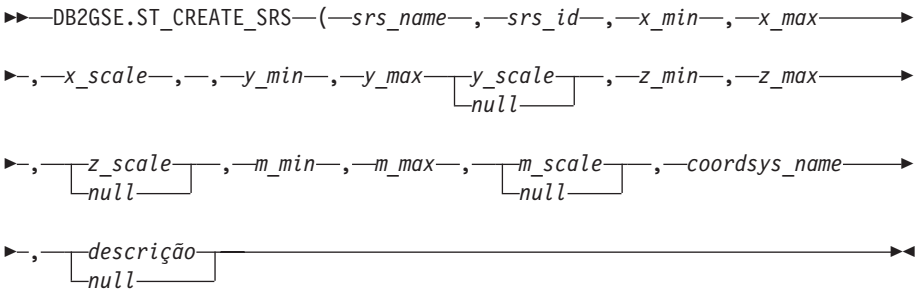
Nenhuma ação é necessária.

Sintaxe

Com fatores de conversão (versão 1)



Com o máximo de extensão possível (versão 2)



Descrições de parâmetros

Com fatores de conversão (versão 1)

srs_name
Identifica o sistema de referência espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *srs_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

srs_id Identifica exclusivamente o sistema de referência espacial. Este

identificador numérico é utilizado como um parâmetro de entrada para várias funções espaciais. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é INTEGER.

x_offset
Especifica o deslocamento de todas as coordenadas X de geometrias que são representadas neste sistema de referência espacial. O deslocamento é subtraído antes do fator de escala *x_scale* ser aplicado quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender. (*WKT* é texto reconhecido e *WKB* é binário reconhecido). Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, um valor 0 (zero) será utilizado.

O tipo de dados deste parâmetro é DOUBLE.

x_scale
Especifica o fator de escala de todas as coordenadas X de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após o deslocamento *x_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender. O valor de *x_offset* é especificado explicitamente ou um valor padrão 0 é utilizado para *x_offset*. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é DOUBLE.

y_offset
Especifica o deslocamento de todas as coordenadas Y de geometrias que são representadas neste sistema de referência espacial. O deslocamento é subtraído antes do fator de escala *y_scale* ser aplicado quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, um valor 0 (zero) será utilizado.

O tipo de dados deste parâmetro é DOUBLE.

y_scale
Especifica o fator de escala de todas as coordenadas Y de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após o deslocamento *y_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender O valor de *y_offset* é especificado explicitamente ou um valor padrão 0 é utilizado para *y_offset*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor do parâmetro *x_scale* será utilizado. Se você especificar um valor diferente de nulo para este parâmetro, o valor especificado deverá corresponder ao valor do parâmetro *x_scale*.

O tipo de dados deste parâmetro é DOUBLE.

z_offset

Especifica o deslocamento de todas as coordenadas Z de geometrias que são representadas neste sistema de referência espacial. O deslocamento é subtraído antes do fator de escala *z_scale* ser aplicado quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, um valor 0 (zero) será utilizado.

O tipo de dados deste parâmetro é DOUBLE.

z_scale

Especifica o fator de escala de todas as coordenadas Z de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após o deslocamento *z_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender. O valor de *z_offset* é especificado explicitamente ou um valor padrão 0 é utilizado para *z_offset*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, um valor 1 será utilizado.

O tipo de dados deste parâmetro é DOUBLE.

m_offset

Especifica o deslocamento de todas as coordenadas M de geometrias que são representadas neste sistema de referência espacial. O deslocamento é subtraído antes do fator de escala *m_scale* ser aplicado quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, um valor 0 (zero) será utilizado.

O tipo de dados deste parâmetro é DOUBLE.

m_scale

Especifica o fator de escala de todas as coordenadas M de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após o deslocamento *m_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender. O valor de *m_offset* é especificado explicitamente ou um valor padrão 0 é utilizado para *m_offset*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, um valor 1 será utilizado.

O tipo de dados deste parâmetro é DOUBLE.

coordsys_name

Identifica exclusivamente o sistema de coordenadas no qual se baseia este sistema de referência espacial. O sistema de coordenadas deve estar listado na exibição ST_COORDINATE_SYSTEMS. Você deve fornecer um valor diferente de nulo para este parâmetro.

O valor de *coordsys_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

descrição
Descreve o sistema de referência espacial, explicando a finalidade do aplicativo. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, nenhuma informação de descrição será gravada.

O tipo de dados deste parâmetro é VARCHAR(256).

Com o máximo de extensão possível (versão 2)

srs_name
Identifica o sistema de referência espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *srs_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

srs_id Identifica exclusivamente o sistema de referência espacial. Este identificador numérico é utilizado como um parâmetro de entrada para várias funções espaciais. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é INTEGER.

x_min Especifica o valor mínimo possível da coordenada X para todas as geometrias que utilizam este sistema de referência espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é DOUBLE.

x_max Especifica o valor máximo possível da coordenada X para todas as geometrias que utilizam este sistema de referência espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

Dependendo do valor de *x_scale*, o valor que é mostrado na exibição DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS pode ser maior do que o valor especificado aqui. O valor da exibição está correto.

O tipo de dados deste parâmetro é DOUBLE.

x_scale
Especifica o fator de escala de todas as coordenadas X de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após o deslocamento *x_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender. O cálculo do deslocamento *x_offset* baseia-se no valor de *x_min*. Você deve fornecer um valor diferente de nulo para este parâmetro.

Se os parâmetros *x_scale* e *y_scale* forem especificados, os valores deverão corresponder.

O tipo de dados deste parâmetro é DOUBLE.

y_min Especifica o valor mínimo possível da coordenada Y para todas as geometrias que utilizam este sistema de referência espacial. Você deve fornecer um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é DOUBLE.

y_max Especifica o valor máximo possível da coordenada Y para todas as geometrias que utilizam este sistema de referência espacial. Você deve fornecer um valor diferente de nulo para este parâmetro.

Dependendo do valor de *y_scale*, o valor que é mostrado na exibição DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS pode ser maior do que o valor especificado aqui. O valor da exibição está correto.

O tipo de dados deste parâmetro é DOUBLE.

y_scale

Especifica o fator de escala de todas as coordenadas Y de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após o deslocamento *y_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender O cálculo do deslocamento *y_offset* baseia-se no valor de *y_min*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor do parâmetro *x_scale* será utilizado. Se os parâmetros *y_scale* e *x_scale* forem especificados, os valores deverão corresponder.

O tipo de dados deste parâmetro é DOUBLE.

z_min Especifica o valor mínimo possível da coordenada Z para todas as geometrias que utilizam este sistema de referência espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é DOUBLE.

z_max Especifica o valor máximo possível da coordenada Z para todas as geometrias que utilizam este sistema de referência espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

Dependendo do valor de *z_scale*, o valor que é mostrado na exibição DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS pode ser maior do que o valor especificado aqui. O valor da exibição está correto.

O tipo de dados deste parâmetro é DOUBLE.

z_scale

Especifica o fator de escala de todas as coordenadas Z de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após o deslocamento *z_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender. O cálculo do deslocamento *z_offset* baseia-se no valor de *z_min*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, um valor 1 será utilizado.

O tipo de dados deste parâmetro é DOUBLE.

m_min
Especifica o valor mínimo possível da coordenada M para todas as geometrias que utilizam este sistema de referência espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é DOUBLE.

m_max
Especifica o valor máximo possível da coordenada M para todas as geometrias que utilizam este sistema de referência espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

Dependendo do valor de *m_scale*, o valor que é mostrado na exibição DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS pode ser maior do que o valor especificado aqui. O valor da exibição está correto.

O tipo de dados deste parâmetro é DOUBLE.

m_scale
Especifica o fator de escala de todas as coordenadas M de geometrias que são representadas neste sistema de referência espacial. O fator de escala é aplicado (multiplicação) após o deslocamento *m_offset* ser subtraído quando as geometrias são convertidas de representações externas (WKT, WKB, forma) para a representação interna do DB2 Spatial Extender. O cálculo do deslocamento *m_offset* baseia-se no valor de *m_min*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, um valor 1 será utilizado.

O tipo de dados deste parâmetro é DOUBLE.

coordsys_name
Identifica exclusivamente o sistema de coordenadas no qual se baseia este sistema de referência espacial. O sistema de coordenadas deve estar listado na exibição ST_COORDINATE_SYSTEMS. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *coordsys_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

descrição
Descreve o sistema de referência espacial, explicando a finalidade do aplicativo. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, nenhuma informação de descrição será gravada.

O tipo de dados deste parâmetro é VARCHAR(256).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_CREATE_SRS. Este exemplo utiliza um comando DB2 CALL para criar um sistema de referência espacial denominado SRSDEMO com os seguintes valores de parâmetros:

- *srs_id*: 1000000
- *x_offset*: -180
- *x_scale*: 1000000
- *y_offset*: -90
- *y_scale*: 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',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_DISABLE_AUTOGEOCODING

Utilize este procedimento armazenado para especificar que o DB2 Spatial Extender deve parar a sincronização de uma coluna geocodificada com sua(s) coluna(s) de codificação geográfica associada(s).

Uma *coluna de codificação geográfica* é utilizada como entrada no geocodificador.

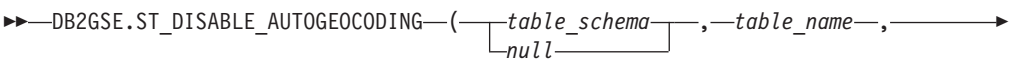
Autorização

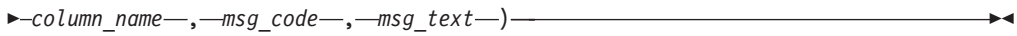
O ID de usuário sob o qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridades DBADM e DATAACCESS no banco de dados que contêm a tabela na qual os acionadores que estão sendo descartados são definidos
- Privilégio CONTROL nesta tabela
- Privilégios ALTER e UPDATE nesta tabela

Nota: Para os privilégios CONTROL e ALTER, é necessário ter autoridade DROPIN no esquema DB2GSE.

Sintaxe





Descrições de parâmetros

table_schema
Nomeia o esquema ao qual pertence a tabela ou exibição especificada no parâmetro *table_name*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor no registro especial CURRENT SCHEMA será utilizado como o nome do esquema para a tabela ou exibição.

O valor de *table_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

table_name
Especifica o nome não qualificado da tabela na qual estão definidos os disparos que você deseja eliminar. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

column_name
Nomeia a coluna geocodificada que é mantida pelos disparos que você deseja eliminar. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_DISABLE_AUTOGEOCODING. Este exemplo utiliza um comando DB2 CALL para desativar a codificação geográfica automática na coluna LOCALIZAÇÃO da tabela denominada CLIENTES:

```
call DB2GSE.ST_DISABLE_AUTOGEOCODING(NULL,'CUSTOMERS','LOCATION',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_DISABLE_DB

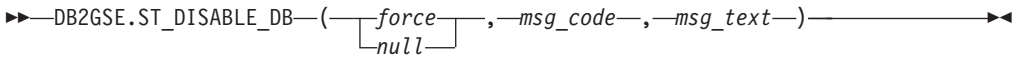
Utilize este procedimento armazenado para remover recursos que permitem ao DB2 Spatial Extender armazenar e suportar dados espaciais.

Este procedimento armazenado ajuda a resolver problemas ou questões que surgem após a ativação do banco de dados para operações espaciais. Por exemplo, você pode ativar um banco de dados para operações espaciais e, depois, decidir utilizar um outro banco de dados com o DB2 Spatial Extender. Contanto que não tenha definido colunas espaciais ou importado dados espaciais, você pode chamar este procedimento armazenado para remover todos os recursos espaciais do primeiro banco de dados. Em razão da interdependência entre colunas espaciais e as definições de tipos, não é possível eliminar as definições de tipos quando existem colunas desses tipos. Se você já tiver definido as colunas espaciais mas ainda desejar desativar um banco de dados para as operações espaciais, deverá especificar um valor diferente de 0 (zero) para o parâmetro *force* para remover todos os recursos espaciais do banco de dados que não possuem outras dependências dos mesmos.

Autorização

O ID do usuário, sob o qual este procedimento armazenado é invocado, deve ter autoridade DBADM no banco de dados a partir do qual os recursos do DB2 Spatial Extender devem ser removidos.

Sintaxe



Descrições de parâmetros

force
Especifica que você deseja desativar um banco de dados para operações espaciais, mesmo que você tenha objetos de banco de dados que sejam dependentes dos tipos espaciais ou funções espaciais. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se você especificar um valor diferente de 0 (zero) ou nulo para o parâmetro *force*, o banco de dados será desativado e todos os recursos do DB2 Spatial Extender serão removidos (se possível). Se você especificar 0 (zero) ou nulo, o banco de dados não será desativado se algum objeto de banco de dados for dependente de tipos espaciais ou funções espaciais. Os objetos de banco de dados que podem ter essas dependências incluem tabelas, exibições, limitações, disparos,

colunas geradas, métodos, funções, procedimentos e outros tipos de dados (subtipos ou tipos estruturados com um atributo espacial).

O tipo de dados deste parâmetro é SMALLINT.

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_DISABLE_DB. Este exemplo utiliza um comando DB2 CALL para desativar o banco de dados para operações espaciais, com o valor 1 para o parâmetro *force*:

```
call DB2GSE.ST_DISABLE_DB(1,?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_DROP_COORDSYS

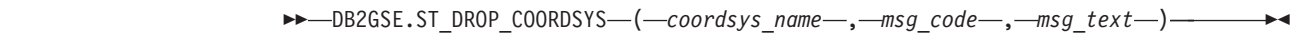
Utilize este procedimento armazenado para excluir informações sobre um sistema de coordenadas do banco de dados. Quando este procedimento armazenado for processado, as informações sobre o sistema de coordenadas não estarão mais disponíveis na visualização de catálogo DB2GSE.ST_COORDINATE_SYSTEMS.

Restrição:
Não é possível eliminar um sistema de coordenadas no qual baseia-se um sistema de referência espacial.

Autorização

O ID do usuário, sob o qual o procedimento armazenado é invocado, deve ter autoridade DBADM.

Sintaxe



Descrições de parâmetros

coordsys_name
Identifica exclusivamente o sistema de coordenadas. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *coordsys_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_DROP_COORDSYS. Este exemplo utiliza um comando DB2 CALL para excluir um sistema de coordenadas denominado NORTH_AMERICAN_TEST do banco de dados:

```
call DB2GSE.ST_DROP_COORDSYS('NORTH_AMERICAN_TEST',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_DROP_SRS

Use este procedimento armazenado para eliminar um sistema de referência espacial.

Quando este procedimento armazenado é processado, as informações sobre o sistema de referência espacial são removidas da exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Restrição: Você não pode eliminar um sistema de referência espacial se uma coluna espacial que o utiliza estiver registrada.

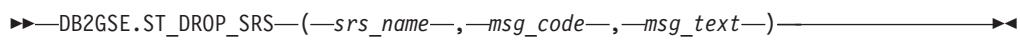
Importante:
Utilize este procedimento armazenado com atenção. Se você utilizar este procedimento armazenado para eliminar um sistema de referência espacial, e

houver dados espaciais associados a esse sistema de referência espacial, não será mais possível executar operações espaciais nos dados espaciais.

Autorização

O ID do usuário, sob o qual o procedimento armazenado é invocado, deve ter autoridade DBADM.

Sintaxe



Descrições de parâmetros

srs_name
Identifica o sistema de referência espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *srs_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_DROP_SRS. Este exemplo utiliza um comando DB2 CALL para excluir um sistema de referência espacial denominado SRSDEMO:

```
call DB2GSE.ST_DROP_SRS('SRSDEMO',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_ENABLE_AUTOGEOCODING

Utilize este procedimento armazenado para especificar que o DB2 Spatial Extender deve sincronizar uma coluna codificada geograficamente com sua(s) coluna(s) de codificação geográfica associada(s).

Uma *coluna de codificação geográfica* é utilizada como entrada no geocodificador. Toda vez que os valores são inseridos ou atualizados na coluna ou colunas de codificação geográfica, os disparos são ativados. Esses disparos chamam o geocodificador associado para geocodificar os valores inseridos ou atualizados e para colocar os dados resultantes na coluna codificada geograficamente.

Restrição: Você só pode ativar a geocodificação automática nas tabelas nas quais os acionadores INSERT e UPDATE podem ser criados. Consequentemente, não é possível ativar o autogeocoding nas exibições ou nos pseudônimos.

Pré-requisito: Antes de ativar a geocodificação automática, você deve executar a etapa de configuração de codificação geográfica chamando o procedimento ST_SETUP_GEOCODING. A etapa de configuração de codificação geográfica especifica os valores de parâmetros de geocodificador e de codificação geográfica. Ela também identifica as colunas de codificação geográfica que devem ser sincronizadas com as colunas geocodificadas.

Autorização

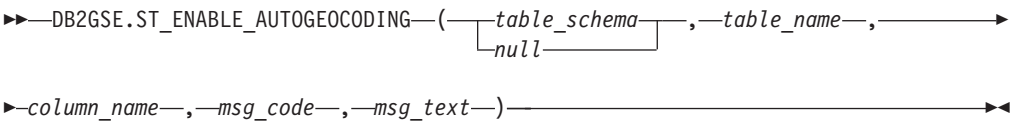
O ID de usuário sob o qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridade DBADM no banco de dados que contém a tabela na qual estão definidos os acionadores que são criados por este procedimento armazenado são definidos
- Privilégio CONTROL sobre a tabela
- Privilégio ALTER na tabela

Se o ID de autorização da instrução não tiver autoridade DBADM, os privilégios que o ID de autorização da instrução mantém (sem considerar os privilégios PUBLIC ou de grupo) deverão incluir todos os seguintes privilégios, desde que exista o acionador:

- Privilégio SELECT na tabela na qual a geocodificação automática ou a autoridade DATAACCESS está ativada
- Privilégios necessários para avaliar as expressões SQL que são especificadas para os parâmetros na configuração de codificação geográfica

Sintaxe



Descrições de parâmetros

table_schema
Identifica o esquema ao qual pertence a tabela especificada no parâmetro *table_name*. Embora você deva especificar um valor para este parâmetro, o

valor pode ser nulo. Se este parâmetro for nulo, o valor no registro especial CURRENT SCHEMA será utilizado como o nome do esquema para a tabela.

O valor de *table_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

table_name
Especifica o nome não qualificado da tabela que contém a coluna na qual os dados geocodificados serão inseridos ou atualizados. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

column_name
Identifica a coluna na qual os dados geocodificados serão inseridos ou atualizados. Esta coluna é chamada de coluna geocodificada. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_ENABLE_AUTOGEOCODING. Este exemplo utiliza um comando DB2 CALL para ativar a geocodificação automática na coluna LOCALIZAÇÃO da tabela denominada CLIENTES:

```
call DB2GSE.ST_ENABLE_AUTOGEOCODING(NULL,'CUSTOMERS','LOCATION',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_ENABLE_DB

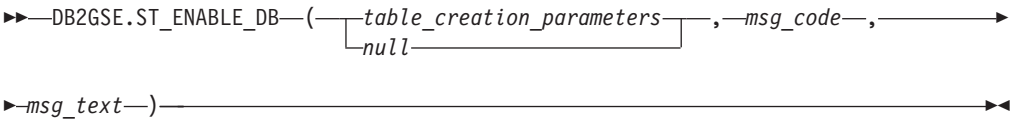
Utilize este procedimento armazenado para fornecer a um banco de dados os recursos de que ele precisa para armazenar dados espaciais e suportar operações espaciais. Estes recursos incluem tipos de dados espaciais, tipos de índices espaciais, exibições do catálogo, funções fornecidas e outros procedimentos armazenados.

Este procedimento armazenado substitui DB2GSE.gse_enable_db.

Autorização

O ID do usuário, sob o qual o procedimento armazenado é invocado, deve ter autoridade DBADM no banco de dados que está sendo ativado.

Sintaxe



Descrições de parâmetros

table_creation_parameters
Especifica quaisquer opções que serão incluídas nas instruções CREATE TABLE das tabelas do catálogo do DB2 Spatial Extender. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, nenhuma opção será incluída nas instruções CREATE TABLE.

Para especificar essas opções, utilize a sintaxe da instrução DB2 CREATE TABLE. Por exemplo, para especificar uma área de tabela na qual as tabelas serão criadas, utilize:

```
IN tsName INDEX IN indexTsName
```

O tipo de dados deste parâmetro é VARCHAR(32 K).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

O exemplo a seguir mostra como usar a Interface do Nível de Chamada (CLI) para chamar o procedimento armazenado 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;

  /* Alocar identificador de ambiente */
  rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);

  /* Alocar identificador de banco de dados */
  rc = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);

  /* Establish a connection to database "testdb" */
  rc = SQLConnect(hdbc, (SQLCHAR *)"testdb", SQL_NTS, (SQLCHAR *)uid,SQL_NTS,
                  (SQLCHAR *)pwd, SQL_NTS);


  /* Alocar identificador de instrução */
  rc = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt) ;

  /* Associar instrução SQL para chamar o procedimento armazenado ST_ENABLE_DB  */
  /* com o identificador de instrução e enviar a instrução para o DBMS para
     ser preparada. */
  rc = SQLPrepare(hstmt, "call DB2GSE!ST_ENABLE_DB(?,?,?)", SQL_NTS);

  /* Ligar o marcador de primeiro parâmetro na instrução de chamada de SQL */
  /* o parâmetro de entrada para os parâmetros de criação de tabela,
     à variável */
  /* 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]);

  /* Ligar o marcador de segundo parâmetro na instrução de chamada de SQL,*/
  /* o parâmetro de saída para código de mensagem retornado, à variável msg_code.      */
  ind[1] = 0;
  rc = SQLBindParameter(hstmt, 2, SQL_PARAM_OUTPUT, SQL_C_LONG,
                        SQL_INTEGER, 0, 0, &msg_code, 4, &ind[1]);

  /* Ligar o marcador de terceiro parâmetro na instrução de
     chamada de SQL, o */
  /* parâmetro de saída para texto da mensagem retornado, à variável 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);
```

Procedimento ST_EXPORT_SHAPE

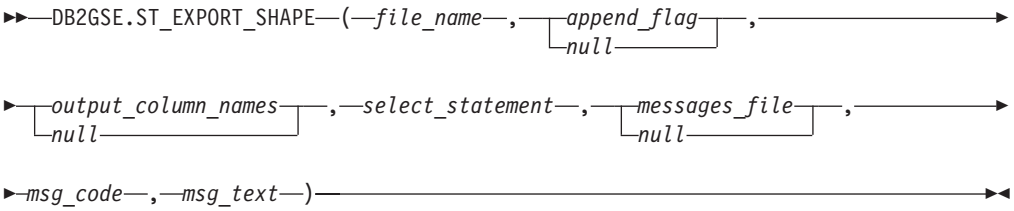
Utilize este procedimento armazenado para exportar uma coluna espacial e sua tabela associada a um arquivo shape.

Autorização

O ID de usuário sob o qual este procedimento armazenado é chamado deve ter os privilégios necessários para executar com êxito a instrução SELECT a partir da qual os dados serão exportados.

O procedimento armazenado, que é executado como um processo que pertence ao proprietário da instância do DB2, deve ter os privilégios necessários na máquina do servidor para criar ou gravar nos arquivos de formas.

Sintaxe



Descrições de parâmetros

file_name
Especifica o nome completo do caminho de um arquivo shape para o qual os dados especificados serão exportados. Você deve especificar um valor diferente de nulo para este parâmetro.

É possível usar o procedimento armazenado ST_EXPORT_SHAPE para exportar um novo arquivo ou para exportar para um arquivo existente, anexando os dados exportados a ele:

- Se você estiver exportando para um novo arquivo, poderá especificar a extensão do arquivo opcional como .shp ou .SHP. Se você especificar .shp ou .SHP para a extensão do arquivo, o DB2 Spatial Extender criará o arquivo com o valor *file_name* especificado. Se você não especificar a extensão do arquivo opcional, o DB2 Spatial Extender criará o arquivo que possui o nome do valor *file_name* especificado e com uma extensão .shp.
- Se você estiver exportando dados anexando os dados a um arquivo existente, o DB2 Spatial Extender primeiro procurará uma correspondência exata do nome especificado para o parâmetro *file_name*. Se o DB2 Spatial Extender não encontrar uma correspondência exata, ele primeiro procurará um arquivo com a extensão .shp e, em seguida, um arquivo com a extensão .SHP.

Se o valor do parâmetro *append_flag* indicar que você não está anexando a um arquivo existente, mas o arquivo nomeado no parâmetro *file_name* já existir, o DB2 Spatial Extender retornará um erro e não irá sobrepor o arquivo.

Consulte as Observações de Uso para obter uma lista de arquivos que são gravados na máquina do servidor. O procedimento armazenado, que é executado como um processo que pertence ao proprietário da instância do DB2, deve ter os privilégios necessários na máquina do servidor para criar ou gravar nos arquivos.

O tipo de dados deste parâmetro é VARCHAR(256).

append_flag
Indica se os dados que devem ser exportados serão anexados a um arquivo

shape existente. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Indique se você deseja anexar a um arquivo shape existente da seguinte maneira:

- Se você desejar anexar dados a um arquivo shape existente, especifique qualquer valor diferente de 0 (zero) e nulo. Nesse caso, a estrutura de arquivos deve corresponder aos dados exportados; caso contrário, um erro será retornado.
- Se você desejar exportar para um novo arquivo, especifique 0 (zero) ou nulo. Nesse caso, o DB2 Spatial Extender não sobrepõe os arquivos existentes.

O tipo de dados deste parâmetro é SMALLINT.

output_column_names
Especifica um ou mais nomes de colunas (separados por vírgulas) que serão utilizados para colunas não espaciais no arquivo dBASE de saída. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, serão utilizados os nomes derivados da instrução SELECT.

Se você especificar este parâmetro, mas não colocar os nomes de colunas entre aspas duplas, os nomes de colunas serão convertidos para maiúsculas. O número de colunas especificadas deve corresponder ao número de colunas retornadas da instrução SELECT, conforme especificado no parâmetro *select_statement*, excluindo a coluna espacial.

O tipo de dados deste parâmetro é VARCHAR(32 K).

select_statement
Especifica a subseleção que retorna os dados a serem exportados. A subseleção deve referir-se exatamente a uma coluna espacial e a qualquer número de colunas de atributo. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é VARCHAR(32 K).

messages_file
Especifica o nome completo do caminho do arquivo (na máquina do servidor) que conterá as mensagens sobre a operação de exportação. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se esse parâmetro for nulo, nenhum arquivo para as mensagens do DB2 Spatial Extender será criado.

As mensagens que são enviadas para este arquivo de mensagens podem ser:

- Mensagens informativas como, por exemplo, um resumo da operação de exportação
- Mensagens de erro de dados que não puderam ser exportados, por exemplo, por causa de sistemas de coordenadas diferentes

O procedimento armazenado, que é executado como um processo que pertence ao proprietário da instância do DB2, deve ter os privilégios necessários no servidor para criar o arquivo.

O tipo de dados deste parâmetro é VARCHAR(256).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou

aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Observações de Uso

Você pode exportar apenas uma coluna espacial por vez.

O procedimento armazenado ST_EXPORT_SHAPE cria ou grava nos quatro arquivos a seguir:

- O arquivo shape principal (extensão .shp).
- O arquivo shape de índice (extensão .shx).
- Um arquivo dBASE que contém dados de colunas não espaciais (extensão .dbf). Este arquivo é criado apenas se as colunas de atributo realmente precisarem ser exportadas
- Um arquivo de projeção que especifica o sistema de coordenadas que está associado aos dados espaciais, se o sistema de coordenadas não for igual a "UNSPECIFIED" (extensão .prj). O sistema de coordenadas é obtido do primeiro registro espacial. Ocorrerá um erro se registros subsequentes tiverem sistemas de coordenadas diferentes.

A tabela a seguir descreve como os tipos de dados DB2 são armazenados nos arquivos de atributos do dBASE. Todos os outros tipos de dados DB2 não são suportados.

Tabela 22. Armazenamento de Tipos de Dados DB2 nos Arquivos de Atributos

Tipo de SQL	Tipo de .dbf	Comprimento do .dbf	Decimais do .dbf	Comentários
SMALLINT	N	6	0	
INTEGER	N	11	0	
BIGINT	N	20	0	
DECIMAL	N	precision+2	escala	
REAL FLOAT(1) até FLOAT(24)	F	14	6	
DOUBLE FLOAT(25) até FLOAT(53)	F	19	9	
CHARACTER, VARCHAR, LONG VARCHAR e DATALINK	C	*len*	0	comprimento ≤ 255
DATE	D	8	0	
TIME	C	8	0	
TIMESTAMP	C	26	0	

Todos os sinônimos de tipos de dados e tipos distintos que baseiam-se nos tipos listados na tabela precedente são suportados.

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_EXPORT_SHAPE. Este exemplo utiliza um comando DB2 CALL para exportar todas as linhas da tabela CUSTOMERS para um arquivo shape que será criado e nomeado `/tmp/export_file`:

```
call DB2GSE.ST_EXPORT_SHAPE('/tmp/export_file',0,NULL,
     'select * from customers','/tmp/export_msg',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_IMPORT_SHAPE

Utilize este procedimento armazenado para importar um arquivo shape para um banco de dados que será ativado para operações espaciais.

O procedimento armazenado pode operar de uma das duas formas, com base no parâmetro *create_table_flag*:

- O DB2 Spatial Extender pode criar uma tabela que possui uma coluna espacial e colunas de atributos e, em seguida, pode carregar as colunas da tabela com os dados do arquivo.
- Caso contrário, os dados de shape e de atributos poderão ser carregados em uma tabela existente que tenha uma coluna espacial e colunas de atributo que correspondam aos dados do arquivo.

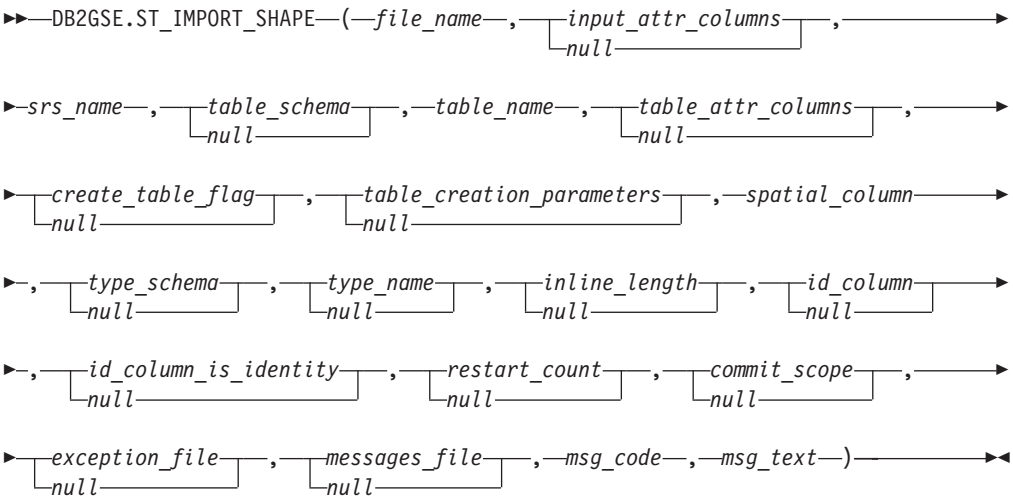
Autorização

O proprietário da instância do DB2 deve ter os privilégios necessários na máquina do servidor para ler os arquivos de entrada e, opcionalmente, gravar os arquivos de erros. Requisitos de autorização adicionais variam dependendo se você está importando para uma tabela existente ou para uma nova tabela.

- **Ao importar para uma tabela existente**, O ID de usuário sob o qual este procedimento armazenado é chamado deve conter uma das seguintes autoridades ou privilégios:
  - DATAACCESS
  - Privilégio CONTROL na tabela ou visualização
  - Privilégio INSERT e SELECT na tabela ou visualização
- **Ao importar para uma nova tabela**, O ID de usuário sob o qual este procedimento armazenado é chamado deve conter uma das seguintes autoridades ou privilégios:
  - DBADM
  - Autoridade CREATETAB no banco de dados

  O ID de usuário também deve ter uma das seguintes autoridades:
  - Autoridade IMPLICIT-SCHEMA no banco de dados, se o nome do esquema da tabela não existir
  - Privilégio CREATEIN no esquema, se o esquema da tabela existir

Sintaxe



Descrições de parâmetros

file_name
Especifica o nome completo do caminho dos arquivos shape que serão importados. Você deve especificar um valor diferente de nulo para este parâmetro.

Se você especificar a extensão do arquivo opcional, especifique .shp ou .SHP. O DB2 Spatial Extender primeiro procura uma correspondência exata do nome do arquivo especificado. Se o DB2 Spatial Extender não encontrar uma correspondência exata, ele primeiro procurará um arquivo com a extensão .shp e, em seguida, um arquivo com a extensão .SHP.

Consulte as Observações de Uso para obter uma lista de arquivos requeridos, que devem residir na máquina do servidor. O procedimento armazenado, que é executado como um processo que pertence ao proprietário da instância do DB2, deve ter os privilégios necessários no servidor para ler os arquivos.

O tipo de dados deste parâmetro é VARCHAR(256).

input_attr_columns
Especifica uma lista de colunas de atributo a serem importadas do arquivo dBASE. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, todas as colunas serão importadas. Se o arquivo dBASE não existir, este parâmetro deverá ser uma sequência vazia ou nulo.

Para especificar um valor diferente de nulo para este parâmetro, utilize uma das seguintes especificações:

- **Liste os nomes de colunas de atributo.** O exemplo a seguir mostra como especificar uma lista dos nomes das colunas de atributos que serão importadas do arquivo dBASE:

  ```
  N(COLUMN1,COLUMN5,COLUMN3,COLUMN7)
  ```

  Se um nome de coluna não for colocado entre aspas duplas, ele será convertido para maiúsculas. Cada nome na lista deve ser separado por uma vírgula. Os nomes resultantes devem corresponder exatamente aos nomes de coluna no arquivo dBASE.
- **Liste os números de colunas de atributo.** O exemplo a seguir mostra como especificar uma lista dos números das colunas de atributos que serão importadas do arquivo dBASE:

```
P(1,5,3,7)
```

As colunas são enumeradas iniciando com 1. Cada número na lista deve ser separado por uma vírgula.

- **Especifique que os dados de atributos não serão importados.** Especifique "", que é uma sequência vazia que especifica explicitamente que o DB2 Spatial Extender *não* importará dados de atributos.

O tipo de dados deste parâmetro é VARCHAR(32 K).

srs_name
Identifica o sistema de referência espacial que será utilizado para as geometrias que são importadas para a coluna espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

A coluna espacial não será registrada. O SRS (sistema de referência espacial) deve existir antes dos dados serem importados. O processo de importação não cria implicitamente o SRS, mas compara o sistema de coordenadas do SRS com o sistema de coordenadas especificado no arquivo .prj (se disponível com o arquivo shape). O processo de importação também verifica se as extensões dos dados no arquivo shape podem ser representadas no sistema de referência espacial especificado. Ou seja, o processo de importação verifica se as extensões estão dentro das coordenadas X, Y, Z e M mínimas e máximas possíveis do SRS.

O valor de *srs_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

table_schema
Nomeia o esquema ao qual pertence a tabela especificada no parâmetro *table_name*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor no registro especial CURRENT SCHEMA será utilizado como o nome do esquema para a tabela ou exibição.

O valor de *table_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

table_name
Especifica o nome não qualificado da tabela na qual o arquivo shape importado será carregado. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

table_attr_columns
Especifica os nomes de colunas da tabela nos quais os dados de atributos do arquivo dBASE serão armazenados. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, os nomes das colunas no arquivo dBASE serão utilizados.

Se este parâmetro for especificado, o número de nomes deverá corresponder ao número de colunas que são importadas do arquivo dBASE. Se a tabela existir, as definições de colunas deverão corresponder aos dados de entrada. Consulte as Observações de Uso para obter uma explicação de como os tipos de dados de atributos são mapeados para os tipos de dados DB2.

O tipo de dados deste parâmetro é VARCHAR(32 K).

create_table_flag
Especifica se o processo de importação criará uma nova tabela. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo ou qualquer outro valor diferente de 0 (zero), uma nova tabela será criada. (Se a tabela já existir, um erro será retornado). Se este parâmetro for 0 (zero), nenhuma tabela será criada, e a tabela já deverá existir.

O tipo de dados deste parâmetro é INTEGER.

table_creation_parameters
Especifica quaisquer opções a serem incluídas na instrução CREATE TABLE que cria uma tabela na qual os dados serão importados. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, nenhuma opção será incluída na instrução CREATE TABLE.

Para especificar quaisquer opções CREATE TABLE, utilize a sintaxe da instrução DB2 CREATE TABLE. Por exemplo, para especificar uma área de tabela na qual as tabelas serão criadas, especifique:

```
IN tsName INDEX IN indexTsName LONG IN longTsName
```

O tipo de dados deste parâmetro é VARCHAR(32 K).

spatial_column
Nome da coluna espacial na tabela para a qual os dados de shape serão carregados. Você deve especificar um valor diferente de nulo para este parâmetro.

Para uma nova tabela, este parâmetro especifica o nome da nova coluna espacial que será criada. Caso contrário, este parâmetro especificará o nome de uma coluna espacial existente na tabela.

O valor de *spatial_column* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

type_schema
Especifica o nome do esquema do tipo de dados espacial (especificado pelo parâmetro *type_name*) que será utilizado ao criar uma coluna espacial em uma nova tabela. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, um valor DB2GSE será utilizado.

O valor de *type_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

type_name
Nomeia o tipo de dados que será utilizado para os valores espaciais. Embora

você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o tipo de dados será determinado pelo arquivo shape e será um dos seguintes tipos:

- ST_Point
- ST_MultiPoint
- ST_MultiLineString
- ST_MultiPolygon

Observe que os arquivos de formas, por definição, permitem uma distinção apenas entre pontos e multipontos, mas não entre polígonos e multipolígonos ou entre sequências de linhas e sequências de linhas múltiplas.

Se você estiver importando para uma tabela que ainda não existe, este tipo de dados também será utilizado para o tipo de dados da coluna espacial. Nesse caso, o tipo de dados também pode ser um supertipo ST_Point, ST_MultiPoint, ST_MultiLineString ou ST_MultiPolygon.

O valor de *type_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

inline_length
Especifica, para uma nova tabela, o número máximo de bytes que serão alocados para a coluna espacial dentro da tabela. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, nenhuma opção INLINE LENGTH explícita será utilizada na instrução CREATE TABLE e os padrões do DB2 serão utilizados implicitamente.

Os registros espaciais que excedem este tamanho são armazenados separadamente na área de tabela LOB, cujo acesso pode ser mais lento.

Os tamanhos típicos que são necessários para vários tipos espaciais são os seguintes:

- **Um ponto:** 292.
- **Multiponto, linha ou polígono:** O maior valor possível. Considere que o número total de bytes em uma linha não deve exceder o limite do tamanho da página da área de tabela para a qual a tabela é criada.

Consulte a documentação do DB2 sobre a instrução SQL CREATE TABLE para uma descrição completa desse valor. Consulte também o utilitário db2dart para determinar o número de geometrias em linha para as tabelas existentes e a capacidade para alterar o comprimento em linha.

O tipo de dados deste parâmetro é INTEGER.

id_column
Nomeia uma coluna que será criada para conter um número exclusivo para cada linha de dados. (As ferramentas ESRI requerem uma coluna denominada SE_ROW_ID). Os valores exclusivos para essa coluna são gerados automaticamente durante o processo de importação. Embora você deva especificar um valor para este parâmetro, o valor poderá ser nulo se nenhuma coluna (com um ID exclusivo em cada linha) existir na tabela ou se você não estiver incluindo essa coluna em uma tabela recentemente criada. Se este parâmetro for nulo, nenhuma coluna será criada ou preenchida com números exclusivos.

Restrição: Você não pode especificar um nome de *id_column* que corresponda ao nome de alguma coluna no arquivo dBASE.

Os requisitos e o efeito deste parâmetro dependem se a tabela já existe.

- **Para uma tabela existente**, o tipo de dados do parâmetro *id_column* poderá ser qualquer tipo inteiro (INTEGER, SMALLINT ou BIGINT).
- **Para uma nova tabela que será criada**, a coluna é incluída na tabela quando o procedimento armazenado criá-la. A coluna será definida da seguinte forma:

  ```
  INTEGER NOT NULL PRIMARY KEY
  ```

  Se o valor do parâmetro *id_column_is_identity* não for nulo e não 0 (zero), a definição será expandida da seguinte forma:

  ```
  INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY
  ( START WITH 1 INCREMENT BY 1 )
  ```

O valor de *id_column* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

id_column_is_identity
Indica se a *id_column* especificada será criada utilizando a cláusula IDENTITY. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for 0 (zero) ou nulo, a coluna não será criada como a coluna de identidade. Se o parâmetro for qualquer valor diferente de 0 ou nulo, a coluna será criada como a coluna de identidade. Este parâmetro será ignorado para tabelas que já existem.

O tipo de dados deste parâmetro é SMALLINT.

restart_count
Especifica que uma operação de importação será iniciada no registro $n + 1$. Os primeiros n registros serão ignorados. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, todos os registros (iniciando com número de registro 1) serão importados.

O tipo de dados deste parâmetro é INTEGER.

commit_scope
Especifica que um COMMIT será executado após pelo menos menos n registros serem importados. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, um valor de 0 (zero) será usado e será executado um COMMIT no final da operação. Isto pode resultar na utilização de um arquivo de log grande e na perda de dados em operações que são interrompidas.

O tipo de dados deste parâmetro é INTEGER.

exception_file
Especifica o nome completo do caminho de um arquivo shape no qual são armazenados os dados de forma que não puderam ser importados. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se o parâmetro for nulo, nenhum arquivo será criado.

Se você especificar um valor para o parâmetro e incluir a extensão do arquivo opcional, especifique .shp ou .SHP. Se a extensão for nula, uma extensão .shp será anexada.

O arquivo de exceção contém o bloco completo de linhas para as quais uma instrução de inserção única falhou. Por exemplo, suponha que uma linha não possa ser importada porque os dados de shape estão incorretamente codificados. Uma instrução de inserção única tenta importar 20 linhas,

incluindo aquela com erro. Devido ao problema com uma única linha, o bloco inteiro de 20 linhas é gravado no arquivo de exceção.

Os registros são gravados no arquivo de exceção apenas quando esses registros podem ser corretamente identificados, como é o caso em que o tipo de registro de formas não é válido. Alguns tipos de danos nos dados de shape (arquivos .shp) e índice de shape (arquivos .shx) não permitem que os registros apropriados sejam identificados. Nesse caso, nenhum registro é gravado no arquivo de exceção e uma mensagem de erro é emitida para relatar o problema.

Se você especificar um valor para este parâmetro, quatro arquivos serão criados na máquina do servidor. Consulte as Observações de Uso para obter uma explicação sobre esses arquivos. O procedimento armazenado, que é executado como um processo que pertence ao proprietário da instância do DB2, deve ter os privilégios necessários no servidor para criar os arquivos. Se os arquivos já existirem, o procedimento armazenado retornará um erro.

O tipo de dados deste parâmetro é VARCHAR(256).

messages_file
Especifica o nome completo do caminho do arquivo (na máquina do servidor) que conterá as mensagens sobre a operação de importação. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se o parâmetro for nulo, nenhum arquivo para as mensagens do DB2 Spatial Extender será criado.

As mensagens que são gravadas no arquivo de mensagens podem ser:

- Mensagens informativas como, por exemplo, um resumo da operação de importação
- Mensagens de erro de dados que não puderam ser importados, por exemplo, por causa de sistemas de coordenadas diferentes

Estas mensagens correspondem aos dados de shape que são armazenados no arquivo de exceção (identificado pelo parâmetro *exception_file*).

O procedimento armazenado, que é executado como um processo que pertence ao proprietário da instância do DB2, deve ter os privilégios necessários no servidor para criar o arquivo. Se o arquivo já existir, o procedimento armazenado retornará um erro.

O tipo de dados deste parâmetro é VARCHAR(256).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Observações de Uso

O procedimento armazenado ST_IMPORT_SHAPE usa de um a quatro arquivos:

- O arquivo shape principal (extensão .shp). Este arquivo é obrigatório.
- O arquivo shape de índice (extensão .shx). Este arquivo é opcional. Se ele existir, o desempenho da operação de importação poderá melhorar.
- Um arquivo dBASE que contém dados de atributo (extensão .dbf). Este arquivo é obrigatório apenas se os dados de atributo vão ser importados.
- O arquivo de projeção que especifica o sistema de coordenadas dos dados de shape (extensão .prj). Este arquivo é opcional. Se ele existir, o sistema de coordenadas definido nele será comparado com o sistema de coordenadas do sistema de referência espacial especificado pelo parâmetro *srs_id*.

A tabela a seguir descreve como os tipos de dados de atributo dBASE são mapeados para os tipos de dados DB2. Todos os outros tipos de dados de atributo não são suportados.

Tabela 23. Relacionamento entre os tipos de dados DB2 e os tipos de dados do atributo dBASE

Tipo .dbf	Comprimento de .dbf (Consulte a nota)	Decimais de .dbf (Consulte a nota)	Tipo SQL	Comentários
N	< 5	0	SMALLINT	
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 tabela inclui as variáveis a seguir, ambas são definidas no cabeçalho do arquivo dBASE:

- *len*, que representa o comprimento total da coluna no arquivo dBASE. O DB2 Spatial Extender utiliza este valor para duas finalidades:
  - Definir a precisão do tipo de dados SQL DECIMAL ou o comprimento do tipo de dados SQL CHAR
  - Determinar qual dos tipos de inteiro ou ponto flutuante devem ser utilizados
- *dec*, que representa o número máximo de dígitos à direita do ponto decimal da coluna no arquivo dBASE. O DB2 Spatial Extender utiliza este valor para definir a escala para o tipo de dados SQL DECIMAL.

Por exemplo, suponha que o arquivo dBASE contenha uma coluna de dados cujo comprimento (*len*) esteja definido como 20. Suponha que o número de dígitos à direita do ponto decimal (*dec*) seja definido como 5. Quando o DB2 Spatial Extender importa dados dessa coluna, ele utiliza os valores de *len* e *dec* para derivar o seguinte tipo de dados SQL: DECIMAL(20,5).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_IMPORT_SHAPE. Este exemplo utiliza um comando DB2 CALL para importar um arquivo shape denominado `/tmp/officesShape` para a tabela 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',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_REGISTER_GEOCODER

Use este procedimento armazenado para registrar um geocodificador.

Pré-requisitos: Antes de registrar um geocodificador:

- Assegure-se de que a função que implementa o geocodificador já esteja criada. Cada função do geocodificador pode ser registrada como um geocodificador com um nome de geocodificador identificado exclusivamente.
- Obtenha informações do fornecedor do geocodificador, tais como:
  - A instrução SQL que cria a função
  - Os valores de parâmetro usados para chamar o procedimento ST_CREATE_SRS para que os dados geométricos possam ser suportados
  - Informações para registrar o geocodificador, tais como:
    - Uma descrição do geocodificador
    - Descrições dos parâmetros para o geocodificador
    - Os valores padrão dos parâmetros de geocodificador

O tipo de retorno da função do geocodificador deve corresponder ao tipo de dados da coluna geocodificada. Os parâmetros de codificação geográfica podem ser um nome de coluna (denominado *coluna de codificação geográfica*) que contém os dados necessários ao geocodificador. Por exemplo, os parâmetros do geocodificador podem identificar endereços ou um valor de significado específico para o geocodificador, tal como o score mínimo de correspondência. Se o parâmetro de codificação geográfica for um nome de coluna, a coluna deverá estar na mesma tabela ou exibição que a coluna geocodificada.

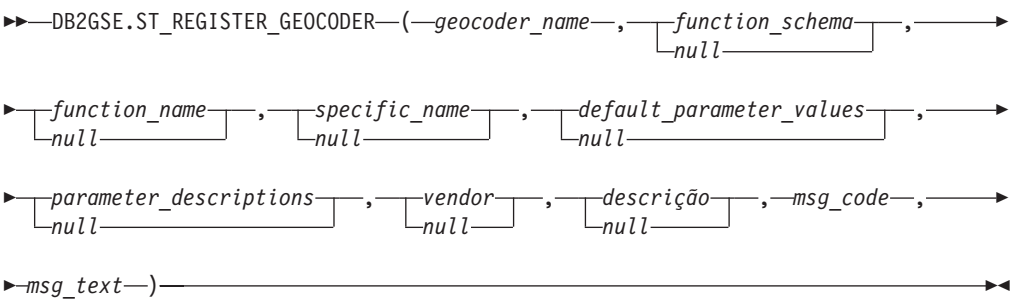
O tipo de retorno da função do geocodificador serve como o tipo de dados para a coluna geocodificada. O tipo de retorno pode ser qualquer tipo de dados DB2, tipo definido pelo usuário ou tipo estruturado. Se um tipo definido pelo usuário ou estruturado for retornado, a função do geocodificador será responsável por retornar um valor válido do respectivo tipo de dados. Se a função do geocodificador retornar valores de um tipo espacial, ou seja, ST_Geometry ou um de seus subtipos, a função do geocodificador será responsável por construir uma geometria válida. A geometria deve ser representada utilizando um sistema de referência espacial existente. A geometria será válida se você chamar a função espacial ST_IsValid na geometria e um valor 1 for retornado. Os dados retornados da função do geocodificador são atualizados ou inseridos na coluna geocodificada, dependendo de qual operação (INSERT ou UPDATE) causou a geração do valor geocodificado.

Para descobrir se um geocodificador já está registrado, examine a exibição do catálogo DB2GSE.ST_GEOCODERS.

Autorização

O ID do usuário, sob o qual esse procedimento armazenado é invocado, deve manter a autoridade DBADM no banco de dados que contém o geocodificador que esse procedimento armazenado registra.

Sintaxe



Descrições de parâmetros

geocoder_name
Identifica exclusivamente o geocodificador. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *geocoder_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

function_schema
Nomeia o esquema para a função que implementa este geocodificador. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor no registro especial CURRENT SCHEMA será utilizado como o nome do esquema para a função.

O valor de *function_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

function_name
Especifica o nome não qualificado da função que implementa este geocodificador. A função já deve estar criada e listada em SYSCAT.ROUTINES.

Para este parâmetro, você pode especificar nulo se o parâmetro *specific_name* for especificado. Se o parâmetro *specific_name* não for especificado, o valor *function_name*, juntamente com o valor *function_schema*, implicitamente ou explicitamente definido, deverá identificar exclusivamente a função. Se o parâmetro *function_name* não for especificado, o DB2 Spatial Extender recuperará o valor *function_name* da visualização de catálogo SYSCAT.ROUTINES.

O valor *function_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

specific_name
Identifica o nome específico da função que implementa o geocodificador. A função já deve estar criada e listada em SYSCAT.ROUTINES.

Para este parâmetro, você pode especificar nulo se o parâmetro *function_name* for especificado e a combinação de *function_schema* e *function_name* identificar exclusivamente a função do geocodificador. Se o nome da função do geocodificador estiver sobrecarregada, o parâmetro *specific_name* não poderá ser nulo. (Um nome de função está *sobrecarregado* se tiver o mesmo nome, mas não os mesmos parâmetros ou tipos de dados de parâmetros, como uma ou mais funções diferentes).

O valor de *specific_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

default_parameter_values
Especifica a lista de valores de parâmetros padrão de codificação geográfica para a função do geocodificador. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se o parâmetro *default_parameter_values* inteiro for nulo, todos os valores de parâmetros padrão serão nulos.

Se você especificar quaisquer valores de parâmetros, especifique-os na ordem em que foram definidos pela função e separe-os com uma vírgula. Por exemplo:

```
default_parm1_value,default_parm2_value,...
```

Cada valor de parâmetro é uma expressão SQL. Siga estas diretrizes:

- Se um valor for uma sequência, coloque-a entre aspas simples.
- Se um valor de parâmetro for um número, não coloque-o entre aspas simples.
- Se o valor do parâmetro for nulo, faça a coerção para o tipo correto. Por exemplo, em vez de especificar apenas NULL, especifique:

  ```
  CAST(NULL AS INTEGER)
  ```

- Se o parâmetro de codificação geográfica for para uma coluna de codificação geográfica, não especifique o valor de parâmetro padrão.

Se algum valor de parâmetro não for especificado (ou seja, se você especificar duas vírgulas consecutivas (`...,,...`)), este parâmetro deverá ser especificado quando a codificação geográfica for configurada ou quando a codificação geográfica for executada no modo em lote com o parâmetro *parameter_values* dos respectivos procedimentos armazenados.

O tipo de dados deste parâmetro é VARCHAR(32 K).

parameter_descriptions
Especifica a lista de descrições de parâmetros de codificação geográfica para a função do geocodificador. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo.

Se o parâmetro *parameter_descriptions* inteiro for nulo, todas as descrições de parâmetros serão nulas. Cada descrição de parâmetro que você especifica explica o significado e uso do parâmetro e pode conter até 256 caracteres. As descrições para os parâmetros devem ser separadas por vírgulas e devem aparecer na ordem dos parâmetros, conforme definido pela função. Se uma

vírgula precisa ser utilizada na descrição de um parâmetro, coloque a sequência entre aspas simples ou duplas. Por exemplo:

```
descrição,'descrição2, que contém uma vírgula',descrição3
```

O tipo de dados deste parâmetro é VARCHAR(32 K).

vendor
Nomeia o fornecedor que implementou o geocodificador. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, nenhuma informação sobre o fornecedor que implementou o geocodificador será gravada.

O tipo de dados deste parâmetro é VARCHAR(128).

descrição
Descreve o geocodificador, explicando sua finalidade. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, nenhuma informação de descrição sobre o geocodificador será gravada.

O tipo de dados deste parâmetro é VARCHAR(256).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo supõe que você deseja criar um geocodificador que utiliza latitude e longitude como entrada e geocodifique para os dados espaciais ST_Point. Para fazer isso, primeiro crie uma função denominada lat_long_gc_func. Em seguida, registre um geocodificador denominado SAMPLEGC, que utiliza a função lat_long_gc_func.

Segue um exemplo da instrução SQL que cria a função lat_long_gc_func, a qual retorna 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)
```

Depois que a função é criada, você pode registrá-la como um geocodificador. Este exemplo mostra como usar o comando CALL do processador de linha de comandos do DB2 para chamar o procedimento armazenado

ST_REGISTER_GEOCODER para registrar um geocodificador chamado SAMPLEGC com a função 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'?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_REGISTER_SPATIAL_COLUMN

Utilize este procedimento armazenado para registrar uma coluna espacial e associar um SRS (sistema de referência espacial) a ele.

Também é possível usar este procedimento armazenado para calcular as extensões geográficas da coluna espacial.

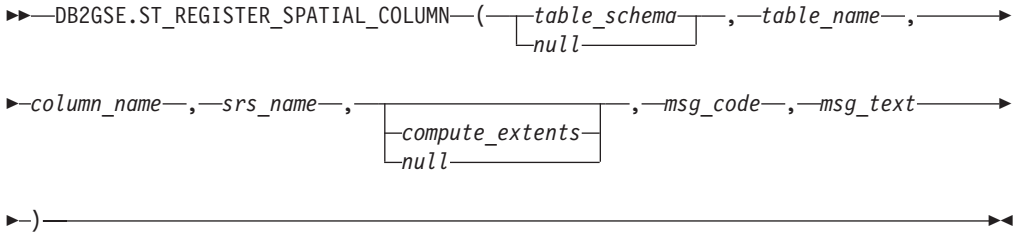
Após o processamento deste procedimento armazenado, as informações sobre a coluna espacial registrada e extensões geográficas ficam disponíveis na visualização de catálogo DB2GSE.ST_GEOMETRY_COLUMNS. Registrar uma coluna espacial cria uma limitação na tabela, se possível, para assegurar que todas as geometrias utilizem o SRS especificado.

Autorização

O ID de usuário com o qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridade DBADM no banco de dados que contém a tabela à qual a coluna espacial que está sendo registrada pertence
- Privilégio CONTROL nesta tabela
- Privilégio ALTER nesta tabela

Sintaxe



Descrições de parâmetros

table_schema

Nomeia o esquema ao qual pertence a tabela ou exibição especificada no parâmetro *table_name*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor no registro especial CURRENT SCHEMA será utilizado como o nome do esquema para a tabela ou exibição.

O valor de *table_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

table_name
Especifica o nome não qualificado da tabela ou exibição que contém a coluna que está sendo registrada. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

column_name
Nomeia a coluna que está sendo registrada. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

srs_name
Nomeia o sistema de referência espacial que será utilizado para esta coluna espacial. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *srs_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

compute_extents
Indica se serão calculadas as extensões geográficas de uma coluna especificada e se serão disponibilizadas por meio da visualização de catálogo DB2GSE.ST_GEOMETRY_COLUMNS. Os possíveis valores para este parâmetro são:

- Um valor maior que 0 para calcular as extensões geográficas.
- Nulo, 0 ou um valor negativo para impedir este cálculo.

Se você omitir este parâmetro, ele terá o mesmo efeito que especificar 0. O cálculo da extensão não é executado.

O tipo de dados deste parâmetro é INTEGER.

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é

retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_REGISTER_SPATIAL_COLUMN. Este exemplo utiliza um comando DB2 CALL para registrar a coluna espacial denominada LOCALIZAÇÃO na tabela denominada CLIENTES. Este comando CALL especifica o valor de parâmetro *srs_name* como USA_SRS_1:

```
call DB2GSE.ST_REGISTER_SPATIAL_COLUMN(NULL,'CUSTOMERS','LOCATION',
     'USA_SRS_1',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_REMOVE_GEOCODING_SETUP

Utilize este procedimento armazenado para remover todas as informações de configuração de codificação geográfica para a coluna geocodificada.

Este procedimento armazenado remove informações que estão associadas à coluna geocodificada especificada a partir das exibições do catálogo DB2GSE.ST_GEOCODING e DB2GSE.ST_GEOCODING_PARAMETERS.

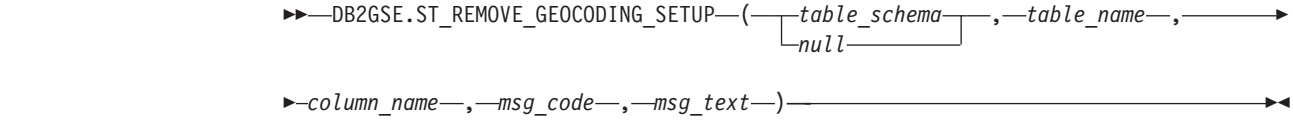
Restrição:
Não é possível remover uma configuração de geocodificação se geocodificação automática estiver ativada para a coluna geocodificada.

Autorização

O ID de usuário com o qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridade DATAACCESS no banco de dados que contém a tabela na qual o geocodificador especificado irá operar
- Privilégio CONTROL nesta tabela
- Privilégio UPDATE nesta tabela

Sintaxe



Descrições de parâmetros

table_schema
Nomeia o esquema ao qual pertence a tabela ou exibição especificada no parâmetro *table_name*. Embora você deva especificar um valor para este

parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor no registro especial CURRENT SCHEMA será utilizado como o nome do esquema para a tabela ou exibição.

O valor de *table_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

table_name
Especifica o nome não qualificado da tabela ou exibição que contém a coluna na qual os dados geocodificados serão inseridos ou atualizados. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

column_name
Nomeia a coluna na qual os dados geocodificados serão inseridos ou atualizados. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados para este parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_REMOVE_GEOCODING_SETUP. Este exemplo utiliza um comando DB2 CALL para remover a configuração de geocodificação da tabela denominada LOCALIZAÇÃO e da coluna LOCALIZAÇÃO:

```
call DB2GSE.ST_REMOVE_GEOCODING_SETUP(NULL, 'CUSTOMERS', 'LOCATION',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_RUN_GEOCODING

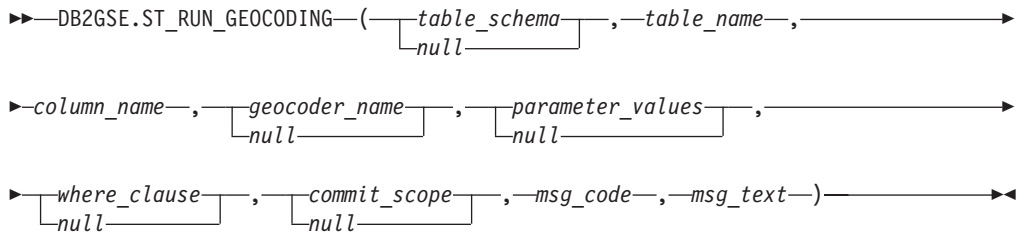
Utilize este procedimento armazenado para executar um geocodificador no modo em lote em uma coluna geocodificada.

Autorização

O ID de usuário com o qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridade DATAACCESS no banco de dados que contém a tabela na qual o geocodificador especificado irá operar
- Privilégio CONTROL nesta tabela
- Privilégio UPDATE nesta tabela

Sintaxe



Descrições de parâmetros

table_schema
Nomeia o esquema ao qual pertence a tabela ou exibição especificada no parâmetro *table_name*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor no registro especial CURRENT SCHEMA será utilizado como o nome do esquema para a tabela ou exibição.

O valor de *table_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

table_name
Especifica o nome não qualificado da tabela ou exibição que contém a coluna na qual os dados geocodificados serão inseridos ou atualizados. Se um nome de exibição for especificado, a exibição deverá ser atualizável. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

column_name
Nomeia a coluna na qual os dados geocodificados serão inseridos ou atualizados. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

geocoder_name
Nomeia o geocodificador que executará a codificação geográfica. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, a codificação geográfica será executada pelo geocodificador que foi especificado quando a codificação geográfica foi configurada.

O valor de *geocoder_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

parameter_values
Especifica a lista de valores de parâmetros de codificação geográfica para a função geocodificador. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se o parâmetro *parameter_values* inteiro for nulo, os valores que são utilizados são os valores de parâmetros que foram especificados quando o geocodificador foi configurado os valores de parâmetros padrão do geocodificador, se o geocodificador não tiver sido configurado.

Se você especificar quaisquer valores de parâmetros, especifique-os na ordem em que foram definidos pela função e separe-os com uma vírgula. Por exemplo:

```
parameter1-value,parameter2-value,...
```

Cada valor de parâmetro pode ser um nome de coluna, uma sequência, um valor numérico ou nulo.

Cada valor de parâmetro é uma expressão SQL. Siga estas diretrizes:

- Se um valor de parâmetro for um nome de coluna de codificação geográfica, assegure-se de que a coluna esteja na mesma tabela ou exibição na qual reside a coluna geocodificada.
- Se um valor de parâmetro for uma sequência, coloque-a entre aspas simples.
- Se um valor de parâmetro for um número, não coloque-o entre aspas simples.
- Se o parâmetro for nulo, faça a coerção para o tipo correto. Por exemplo, em vez de especificar apenas NULL, especifique:

  ```
  CAST(NULL AS INTEGER)
  ```

Se algum valor de parâmetro não for especificado (ou seja, se você especificar duas vírgulas consecutivas (`...,,...`)), este parâmetro deverá ser especificado quando a codificação geográfica for configurada ou quando a codificação geográfica for executada no modo em lote com o parâmetro *parameter_values* dos respectivos procedimentos armazenados.

O tipo de dados deste parâmetro é VARCHAR(32 K).

where_clause
Especifica o conteúdo da cláusula WHERE, que define uma restrição do conjunto de registros que serão geocodificados. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo.

Se o parâmetro *where_clause* for nulo, o comportamento resultante dependerá se a codificação geográfica foi configurada para a coluna (especificada no parâmetro *column_name*) antes da execução do procedimento armazenado. Se o parâmetro *where_clause* for nulo, e:

- Um valor foi especificado quando a codificação geográfica foi configurada, esse valor será utilizado para o parâmetro *where_clause*.
- A codificação geográfica não foi configurada ou nenhum valor foi especificado quando a codificação geográfica foi configurada, ou nenhuma cláusula foi utilizada.

Você pode especificar uma cláusula que faz referência a qualquer coluna na tabela ou exibição em que o geocodificador irá operar. Não especifique a palavra-chave WHERE.

O tipo de dados deste parâmetro é VARCHAR(32 K).

commit_scope
Especifica que um COMMIT será executado após cada *n* registros que são geocodificados. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo.

Se o parâmetro *commit_scope* for nulo, o comportamento resultante dependerá se a codificação geográfica foi configurada para a coluna (especificado no parâmetro *column_name*) antes da execução do procedimento armazenado. Se o parâmetro *commit_scope* for nulo e:

- Um valor foi especificado quando a codificação geográfica foi configurada para a coluna, esse valor será utilizado para o parâmetro *commit_scope*.
- A codificação geográfica não foi configurada ou foi configurada mas nenhum valor foi especificado, o valor padrão de 0 (zero) será usado e será executado um COMMIT no final da operação.

O tipo de dados deste parâmetro é INTEGER.

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_RUN_GEOCODING. Este exemplo utiliza um comando DB2 CALL para geocodificar a coluna LOCALIZAÇÃO na tabela denominada CLIENTES. Este comando CALL especifica o valor do parâmetro *geocoder_name* como SAMPLEGC e o valor do parâmetro *commit_scope* como 10. Um COMMIT será executado após cada 10 registros serem geocodificados:

```
call DB2GSE.ST_RUN_GEOCODING(NULL, 'CUSTOMERS', 'LOCATION',
     'SAMPLEGC',NULL,NULL,10,?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_SETUP_GEOCODING

Utilize este procedimento armazenado para associar uma coluna que será geocodificada a um geocodificador e para configurar os parâmetros de codificação geográfica correspondentes.

Informações sobre a configuração estão disponíveis por meio das visualizações de catálogo DB2GSE.ST_GEOCODING e DB2GSE.ST_GEOCODING_PARAMETERS.

Este procedimento armazenado não chama a codificação geográfica. Ele fornece um modo para especificar as definições de parâmetros para a coluna que será geocodificada. Com essas definições, a chamada subsequente de codificação geográfica em lote ou geocodificação automática pode ser feita com uma interface muito mais simples. As definições de parâmetros que são especificadas nesta etapa de configuração substituem qualquer um dos valores de parâmetros padrão do geocodificador que foram especificados quando o geocodificador foi registrado. Também é possível substituir estas configurações de parâmetros executando o procedimento ST_RUN_GEOCODING no modo em lote.

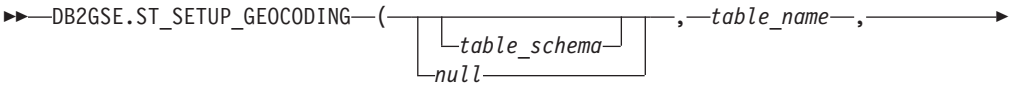
Esta etapa é um pré-requisito para autogeocoding. Você não pode ativar o autogeocoding sem antes configurar os parâmetros de codificação geográfica. Esta etapa não é um pré-requisito para a codificação geográfica em lote. Você pode executar a codificação geográfica no modo em lote com ou sem executar a etapa de configuração. No entanto, se a etapa de configuração for feita antes da codificação geográfica em lote, os valores de parâmetro serão extraídos do tempo de configuração se não forem especificados no tempo de execução.

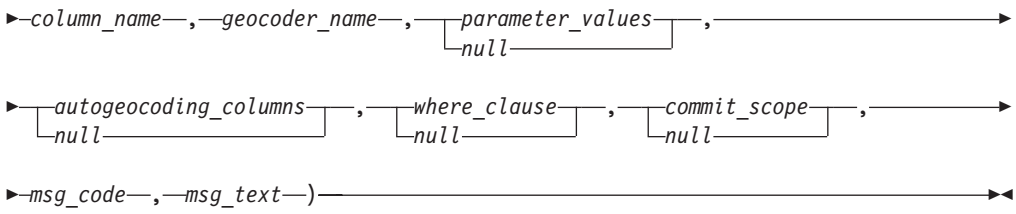
Autorização

O ID de usuário com o qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridade DATAACCESS no banco de dados que contém a tabela na qual o geocodificador especificado irá operar
- Privilégio CONTROL nesta tabela
- Privilégio UPDATE nesta tabela

Sintaxe





Descrições de parâmetros

table_schema
Nomeia o esquema ao qual pertence a tabela ou exibição especificada no parâmetro *table_name*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor no registro especial CURRENT SCHEMA será utilizado como o nome do esquema para a tabela ou exibição.

O valor de *table_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

table_name
Especifica o nome não qualificado da tabela ou exibição que contém a coluna na qual os dados geocodificados serão inseridos ou atualizados. Se um nome de exibição for especificado, a exibição deverá ser atualizável. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

column_name
Nomeia a coluna na qual os dados geocodificados serão inseridos ou atualizados. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

geocoder_name
Nomeia o geocodificador que executará a codificação geográfica. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *geocoder_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

parameter_values
Especifica a lista de valores de parâmetros de codificação geográfica para a função geocodificador. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se o parâmetro *parameter_values* inteiro for nulo, os valores utilizados são extraídos dos valores de parâmetros padrão no momento em que o geocodificador foi registrado.

Se você especificar valores de parâmetros, especifique-os na ordem em que a função os definiu e separe-os com uma vírgula. Por exemplo:

```
parameter1-value,parameter2-value,...
```

Cada valor de parâmetro é uma expressão SQL e pode ser um nome de coluna, uma sequência, um valor numérico ou nulo. Siga estas diretrizes:

- Se um valor de parâmetro for um nome de coluna de codificação geográfica, assegure-se de que a coluna esteja na mesma tabela ou exibição na qual reside a coluna geocodificada.
- Se um valor de parâmetro for uma sequência, coloque-a entre aspas simples.
- Se um valor de parâmetro for um número, não coloque-o entre aspas simples.
- Se o valor de parâmetro for especificado como um valor nulo, faça a coerção para o tipo correto. Por exemplo, em vez de especificar apenas NULL, especifique:

  ```
  CAST(NULL AS INTEGER)
  ```

Se algum valor de parâmetro não for especificado (ou seja, se você especificar duas vírgulas consecutivas (`...,,...`)), este parâmetro deverá ser especificado quando a codificação geográfica for configurada ou quando a codificação geográfica for executada no modo em lote com o parâmetro *parameter_values* dos respectivos procedimentos armazenados.

O tipo de dados deste parâmetro é VARCHAR(32 K).

autogeocoding_columns
Especifica a lista de nomes de colunas nos quais o disparo será criado. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo e o autogeocoding estiver ativado, uma atualização de qualquer coluna na tabela ativará o disparo.

Se você especificar um valor para o parâmetro *autogeocoding_columns*, especifique os nomes de colunas em qualquer ordem e separe-os com uma vírgula. O nome da coluna deve existir na mesma tabela na qual a coluna geocodificada reside.

Esta definição de parâmetro aplica-se apenas ao autogeocoding subsequente.

O tipo de dados deste parâmetro é VARCHAR(32 K).

where_clause
Especifica o conteúdo da cláusula WHERE, que define uma restrição do conjunto de registros que serão geocodificados. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, nenhuma restrição será definida na cláusula WHERE.

A cláusula pode referenciar qualquer coluna na tabela ou exibição na qual o geocodificador irá operar. Não especifique a palavra-chave WHERE.

Esta definição de parâmetro aplica-se apenas à codificação geográfica de modo em lote subsequente.

O tipo de dados deste parâmetro é VARCHAR(32 K).

commit_scope
Especifica que um COMMIT será executado para cada *n* registros que serão geocodificados. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, um COMMIT será executado após todos os registros serem geocodificados.

Esta definição de parâmetro aplica-se apenas à codificação geográfica de modo em lote subsequente.

O tipo de dados deste parâmetro é INTEGER.

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_SETUP_GEOCODING. Este exemplo utiliza um comando DB2 CALL para configurar um processo de geocodificação para a coluna geocodificada denominada LOCALIZAÇÃO na tabela CLIENTES. Este comando CALL especifica o valor de parâmetro *geocoder_name* como SAMPLEGC:

```
call DB2GSE.ST_SETUP_GEOCODING(NULL, 'CUSTOMERS', 'LOCATION',
    'SAMPLEGC','ADDRESS,CITY,STATE,ZIP,1,100,80,,,,"$HOME/sqllib/
     gse/refdata/ky.edg","$HOME/sqllib/samples/extenders/spatial/EDGESample.loc"',
     'ENDEREÇO,CIDADE,ESTADO,CEP',NULL,10,?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado

Procedimento ST_UNREGISTER_GEOCODER

Use este procedimento armazenado para cancelar o registro de um geocodificador.

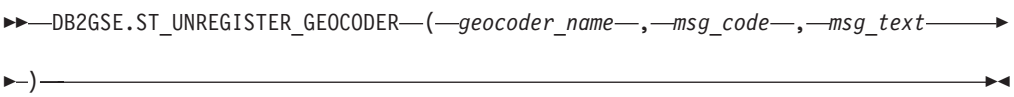
Restrição:
Não é possível cancelar o registro de um geocodificador se ele for especificado na configuração de geocodificação de alguma coluna.

Para determinar se um geocodificador está especificado na configuração de codificação geográfica de uma coluna, verifique as exibições do catálogo DB2GSE.ST_GEOCODING e DB2GSE.ST_GEOCODING_PARAMETERS. Para procurar informações sobre o geocodificador que você deseja cancelar o registro, consulte a exibição do catálogo DB2GSE.ST_GEOCODERS.

Autorização

O ID do usuário, sob o qual este procedimento armazenado é invocado, deve manter a autoridade DBADM no banco de dados que contém o geocodificador cujo registro deve ser cancelado.

Sintaxe



Descrições de parâmetros

geocoder_name
Identifica exclusivamente o geocodificador. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *geocoder_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_UNREGISTER_GEOCODER. Este exemplo utiliza um comando DB2 CALL para cancelar o registro do geocodificador denominado SAMPLEGC:

```
call DB2GSE.ST_UNREGISTER_GEOCODER('SAMPLEGC',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Procedimento ST_UNREGISTER_SPATIAL_COLUMN

Utilize este procedimento armazenado para remover o registro de uma coluna espacial.

O procedimento armazenado remove o registro da seguinte forma:

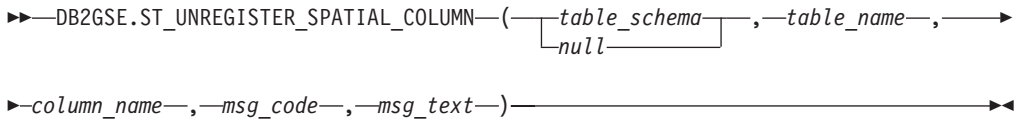
- Removendo a associação do sistema de referência espacial com a coluna espacial. A exibição do catálogo ST_GEOMETRY_COLUMNS continua contendo a coluna espacial, mas a coluna não está mais associada a nenhum sistema de referência espacial.
- Para uma tabela base, eliminando a limitação que o DB2 Spatial Extender colocou nesta tabela para assegurar que os valores de geometria nesta coluna espacial sejam todos representados no mesmo sistema de referência espacial.

Autorização

O ID de usuário com o qual este procedimento armazenado é chamado deve conter um dos seguintes privilégios ou autoridades:

- Autoridade DBADM
- Privilégio CONTROL nesta tabela
- Privilégio ALTER nesta tabela

Sintaxe



Descrições de parâmetros

table_schema
Nomeia o esquema ao qual pertence a tabela especificada no parâmetro *table_name*. Embora você deva especificar um valor para este parâmetro, o valor pode ser nulo. Se este parâmetro for nulo, o valor no registro especial CURRENT SCHEMA será utilizado como o nome do esquema para a tabela ou exibição.

O valor de *table_schema* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

table_name
Especifica o nome não qualificado da tabela que contém a coluna especificada no parâmetro *column_name*. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *table_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

column_name
Nomeia a coluna espacial que você deseja cancelar o registro. Você deve especificar um valor diferente de nulo para este parâmetro.

O valor de *column_name* é convertido para maiúsculas, a menos que seja colocado entre aspas duplas.

O tipo de dados deste parâmetro é VARCHAR(128) ou, se você colocar o valor entre aspas duplas, VARCHAR(130).

Parâmetros de saída

msg_code
Especifica o código de mensagem que é retornado do procedimento armazenado. O valor deste parâmetro de saída identifica a condição de erro, sucesso ou aviso que foi encontrada durante o processamento do procedimento. Se este valor de parâmetro for para uma condição de sucesso ou aviso, o procedimento concluiu sua tarefa. Se o valor de parâmetro for para uma condição de erro, nenhuma alteração foi executada no banco de dados.

O tipo de dados deste parâmetro de saída é INTEGER.

msg_text
Especifica o texto da mensagem real, associado ao código de mensagem, que é retornado do procedimento armazenado. O texto da mensagem pode incluir informações adicionais sobre a condição de sucesso, aviso ou erro, tal como onde um erro foi encontrado.

O tipo de dados deste parâmetro de saída é VARCHAR(1024).

Exemplo

Este exemplo mostra como usar o processador de linha de comandos do DB2 para chamar o procedimento armazenado ST_UNREGISTER_SPATIAL_COLUMN. Este exemplo utiliza um comando DB2 CALL para cancelar o registro da coluna espacial denominada LOCALIZAÇÃO na tabela CLIENTES:

```
call DB2GSE.ST_UNREGISTER_SPATIAL_COLUMN(NULL,'CUSTOMERS','LOCATION',?,?)
```

Os dois pontos de interrogação no final deste comando CALL representam os parâmetros de saída, *msg_code* e *msg_text*. Os valores para estes parâmetros de saída são exibidos após a execução do procedimento armazenado.

Capítulo 18. Funções Espaciais

Use funções espaciais para programar seus aplicativos. Aprenda sobre os fatores que são comuns a todas ou à maioria das funções espaciais e o uso da função por categoria.

Considerações e Tipos de Dados Associados para Funções Espaciais

Esta seção fornece informações necessárias para a codificação de funções espaciais.

Essas informações incluem:

- Fatores a serem considerados: os requisitos para especificar o esquema ao qual as funções espaciais pertencem e o fato de que algumas funções podem ser chamadas como métodos.
- Como tratar uma situação na qual uma função espacial não pode processar o tipo de geometria retornada por outra função espacial.
- Uma tabela mostrando quais funções assumem valores de cada tipo de dados espacial como entrada.

Ao utilizar funções espaciais, esteja ciente destes fatores:

- Antes de chamar uma função espacial, seu nome deve ser qualificado pelo nome do esquema ao qual as funções espaciais pertencem: DB2GSE. Uma forma de fazer isto é especificar explicitamente o esquema na instrução SQL que faz referência à função, por exemplo:

  ```
  SELECT db2gse.ST_Relate (g1, g2, 'T*F**FFF2') EQUALS FROM relate_test
  ```

  Como opção, para evitar especificar o esquema sempre que uma função deve ser chamada, você pode incluir DB2GSE no registro especial CURRENT FUNCTION PATH. Para obter as definições atuais desse registro especial, digite o seguinte comando SQL:

  ```
  VALUES CURRENT FUNCTION PATH
  ```

  Para atualizar o registro especial CURRENT FUNCTION com DB2GSE, emita o seguinte comando SQL:

  ```
  set CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
  ```

- Algumas funções espaciais podem ser chamadas como métodos. No código a seguir, por exemplo, ST_Area é chamado primeiro como função e depois como um método. Nos dois casos, ST_Area é codificado para operar em um polígono que tem ID 10 e que é armazenado na coluna SALES_ZONE de uma tabela denominada STORES. Quando chamado, o ST_Area retornará a área do recurso do mundo real — Zona de Vendas nº 10 — que o polígono representa.

  ST_Area chamou como função:

  ```
  SELECT ST_Area(sales_zone)
  FROM   stores
  WHERE  id = 10
  ```

  ST_Area chamou como um método:

  ```
  SELECT sales_zone..ST_Area()
  FROM   stores
  WHERE  id = 10
  ```

As funções ST_BuildMBRAggr e ST_BuildUnionAggr são descritas em "MBR Aggregate" e "Union Aggregate", respectivamente.

Tratando Valores de ST_Geometry como Valores de um Subtipo

Se uma função espacial retorna uma geometria cujo tipo estático é um tipo super e se a geometria for transmitida para uma função que aceita somente geometrias de um tipo subordinado a esse tipo super, surgirá uma exceção de tempo de compilação.

Por exemplo, o tipo estático do parâmetro de saída da função ST_Union é ST_Geometry, o tipo super de todos os tipos de dados espaciais. O parâmetro de entrada estático da função ST_PointOnSurface pode ser ST_Polygon ou ST_MultiPolygon, dois subtipos de ST_Geometry. Se o DB2 Spatial Extender tentar transmitir geometrias retornadas por ST_Union para ST_PointOnSurface, o DB2 Spatial Extender emitirá a seguinte exceção de tempo de compilação:

```
SQL00440N Não foi localizada função com o nome "ST_POINTONSURFACE"
com argumentos compatíveis no caminho da
função.    SQLSTATE=42884
```

Esta mensagem indica que o DB2 Spatial Extender não pôde localizar uma função chamada ST_PointOnSurface e que possui um parâmetro de entrada de ST_Geometry.

Para permitir que as geometrias do tipo super transmitam funções que aceitem somente subtipos do tipo super, utilize o operador TREAT. Conforme indicado anteriormente, ST_Union retorna geometrias de um tipo estático ST_Geometry. Também podem ser retornadas geometrias de um subtipo dinâmico de ST_Geometry. Suponha, por exemplo, que seja retornada uma geometria com um tipo dinâmico ST_MultiPolygon. Nesse caso, o operador TREAT requer que essa geometria seja utilizada com o tipo estático ST_MultiPolygon. Isto corresponde a um dos tipos de dados do parâmetro de entrada ST_PointOnSurface. Se ST_Union não retornar um valor ST_MultiPolygon, o DB2 Spatial Extender emitirá uma exceção de tempo de execução.

Se uma função retornar uma geometria de um supertipo, o operador TREAT geralmente pode instruir o DB2 Spatial Extender a considerar esta geometria como um subtipo deste supertipo. Mas esteja ciente de que essa operação é bem-sucedida somente se o subtipo corresponder ou for subordinado a um subtipo estático definido como um parâmetro de entrada da função para a qual a geometria é transmitida. Se esta condição não for atendida, o DB2 Spatial Extender emitirá uma exceção de tempo de execução.

Considere outro exemplo: suponha que você deseje determinar os pontos perpendiculares para um determinado ponto no limite de um polígono que não tem orifícios. Utilize a função ST_Boundary para derivar o limite do polígono. O parâmetro de saída estático de ST_Boundary é ST_Geometry, mas ST_PerpPoints aceita geometrias ST_Curve. Como todos os polígonos têm uma sequência de linha (que também é uma curva) como limite e como o tipo de dados das sequências de linhas (ST_LineString) é subordinado a ST_Curve, a operação a seguir permite que um polígono ST_Geometry retornado por ST_Boundary seja transmitido para ST_PerpPoints:

```
SELECT ST_AsText(ST_PerpPoints(TREAT(ST_Boundary(polygon) as ST_Curve)),
       ST_Point(30.5, 65.3, 1)))
FROM   polygon_table
```

Em vez de chamar ST_Boundary e ST_PerpPoints como funções, você pode chamá-los como métodos. Para isto, especifique o seguinte código:

```
SELECT TREAT(ST_Boundary(polygon) as ST_Curve)..
       ST_PerpPoints(St_Point(30.5, 65.3, ))..ST_AsText()
FROM   polygon_table
```

Funções Espaciais de Acordo com o Tipo de Entrada

Lista de funções espaciais de acordo com o tipo de entrada que elas aceitam.

Importante: Como indicado em outro local, os tipos de dados espaciais formam uma hierarquia, com ST_Geometry como raiz. Quando a documentação do DB2 Spatial Extender indica que um valor do tipo super nessa hierarquia pode ser utilizado como entrada para uma função, como opção, um valor de qualquer subtipo desse tipo super também pode ser utilizado como entrada para a função.

Por exemplo, as primeiras entradas na Tabela 24 indicam que ST_Area e diversas outras funções podem assumir valores do tipo de dados ST_Geometry como entrada. Portanto, a entrada dessas funções também pode ser valores de qualquer subtipo de ST_Geometry: ST_Point, ST_Curve, ST_LineString, etc.

Tabela 24. Funções espaciais relacionadas de acordo com o tipo de entrada

Tipo de dados do parâmetro de entrada	Função
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 ou ST_LocateAlong
	ST_Generalize
	ST_GeometryType

Tabela 24. Funções espaciais relacionadas de acordo com o tipo de entrada (continuação)

Tipo de dados do parâmetro de entrada	Função
ST_Geometry, continuação	
	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 ou ST_LocateBetween
	ST_MinM
	ST_MinX
	ST_MinY
	ST_MinZ
	ST_NumPoints
	ST_Overlaps
	ST_Relate
	ST_SRID ou 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

Tabela 24. Funções espaciais relacionadas de acordo com o tipo de entrada (continuação)

Tipo de dados do parâmetro de entrada	Função
ST_Surface	
	ST_Perimeter
	ST_PointOnSurface
ST_GeomCollection	
	ST_GeometryN
	ST_NumGeometries
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

Categorias e Usos para Funções Espaciais

Esta seção apresenta todas as funções espaciais e as organiza por categoria.

O DB2 Spatial Extender fornece funções que:

- Convertem geometrias de e para vários formatos de troca de dados. Essas funções são chamadas de funções construtoras.
- Comparam geometrias para limites, interseções e outras informações. Essas funções são chamadas de funções de comparação.
- Retornam informações sobre as propriedades das geometrias, como coordenadas e medidas nas geometrias, relações entre as geometrias, além de limite e outras informações.
- Geram novas geometrias a partir de geometrias existentes.
- Medem a distância mais curta entre os pontos de geometrias.
- Fornecem informações sobre os parâmetros de índices.
- Fornecem projeções e conversões entre diferentes sistemas de coordenadas.

Funções Construtoras para Converter em e a partir de Formatos de Troca de Dados

O DB2 Spatial Extender fornece funções espaciais que convertem geometrias para e a partir dos formatos de troca de dados.

Os formatos de troca de dados suportados são:

- Representação WKT (well-known text)
- Representação WKB (well-known binary)
- Representação de ESRI shape
- Representação de GML (Geography Markup Language)

As funções para a criação de geometrias a partir desses são conhecidas como *funções construtoras*. As funções construtoras possuem o mesmo nome do tipo de dados de geometria da coluna em que os dados serão inseridos. Essas funções operam de forma consistente em cada um dos formatos de troca de dados de entrada. Esta seção fornece:

- A SQL para chamar funções que operam em formatos de troca de dados e o tipo de geometria retornado por essas funções
- A SQL para chamar uma função que cria pontos a partir das coordenadas X e Y e o tipo de geometria retornado por essa função
- Exemplos de código e conjuntos de resultados

Funções que Convertem de Formatos de Troca de Dados em Geometrias

A conversão de representações de geometria de formatos de troca de dados em valores de geometria permite que representações sejam trocadas como valores de geometria.

Funções que Convertem de Geometrias em Representação Well-known Text (WKT)

Converter geometrias em representações de WKT permite que as geometrias sejam trocadas no formato de texto ASCII. As representações de WKT são valores CLOB que representam sequências de caracteres ASCII.

A função **ST_AsText** converte um valor de geometria armazenado em uma tabela em uma sequência WKT. O exemplo a seguir utiliza uma consulta simples de linha de comando para selecionar os valores que foram inseridos anteriormente na tabela 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)
```

O exemplo a seguir utiliza SQL incorporada para selecionar os valores que foram inseridos anteriormente na tabela 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;
```

Alternativamente, você pode utilizar o grupo de transformação ST_WellKnownText para converter geometrias em sua representação de texto reconhecido ao ligá-los. O código de amostra a seguir mostra como utilizar o grupo de transformação.

```
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;
```

Não são utilizadas funções espaciais na instrução SELECT para converter a geometria.

Além das funções explicadas nesta seção, o DB2 Spatial Extender fornece outras funções que também convertem geometrias em e a partir de representações well-known text. O DB2 Spatial Extender fornece essas outras funções para implementar a especificação "Recursos Simples para SQL" do OGC e a Parte 3: Padrão espacial de ISO SQL/MM. Estas funções são:

- **ST_WKTToSQL**
- **ST_GeomFromText**
- **ST_GeomCollFromTxt**
- **ST_PointFromText**
- **ST_LineFromText**
- **ST_PolyFromText**
- **ST_MPointFromText**
- **ST_MLineFromText**
- **ST_MPolyFromText**

Funções que Convertem de Geometrias em Representação Binária Reconhecida (WKB)

Converter geometrias em representações de WKB permite que as geometrias sejam trocadas no formato binário. A representação de WKB consiste em estruturas de dados binários que devem ser valores BLOB. Esses valores BLOB representam estruturas de dados binários que devem ser gerenciadas por um programa aplicativo escrito em uma linguagem de programação que o DB2 suporta e para a qual o DB2 tenha uma ligação de idioma.

A função **ST_AsBinary** converte um valor de geometria armazenado em uma tabela na representação de WKB (binário reconhecido), que pode ser buscada em uma variável BLOB no armazenamento do programa. O exemplo a seguir utiliza SQL incorporada para selecionar os valores que foram inseridos anteriormente na tabela 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;
```

Alternativamente, você pode utilizar o grupo de transformação ST_WellKnownBinary para converter geometrias em sua representação de binário reconhecido ao ligá-los. O código de amostra a seguir mostra como utilizar esse grupo de transformação.

```
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;
```

Não são utilizadas funções espaciais na instrução SELECT para converter a geometria.

Além das funções explicadas nesta seção, existem outras funções que também convertem geometrias de e para representações de binário reconhecido. O DB2 Spatial Extender fornece essas outras funções para implementar a especificação "Recursos Simples para SQL" do OGC e a Parte 3: Padrão espacial de ISO SQL/MM. Estas funções são:

- **ST_WKBToSQL**
- **ST_GeomFromWKB**
- **ST_GeomCollFromWKB**
- **ST_PointFromWKB**
- **ST_LineFromWKB**
- **ST_PolyFromWKB**
- **ST_MPointFromWKB**
- **ST_MLineFromWKB**
- **ST_MPolyFromWKB**

Funções que Convertem de Geometrias em Representação de Forma ESRI

Converter geometrias em representações de formato ESRI permite que as geometrias sejam trocadas no formato binário. A representação de Formato ESRI consiste em estruturas de dados binários que devem ser gerenciadas por um programa aplicativo escrito em uma linguagem suportada.

A função **ST_AsShape** converte um valor de geometria armazenado em uma tabela na representação de Formato ESRI, que pode ser buscada em uma variável BLOB no armazenamento do programa. O exemplo a seguir utiliza SQL incorporada para selecionar os valores que foram inseridos anteriormente na tabela 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;
```

Alternativamente, você pode utilizar o grupo de transformação ST_Shape para converter geometrias implicitamente em sua representação de forma ao ligá-las. O código de amostra a seguir mostra como utilizar o grupo de transformação.

```
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;
```

Não são utilizadas funções espaciais na instrução SELECT para converter a geometria.

Funções que Convertem de Geometrias em Representação Geography Markup Language (GML)

Converter geometrias em representações de GML permite que as geometrias sejam trocadas no formato de texto ASCII. Representações de GML são sequências ASCII.

A função **ST_AsGML** converte um valor de geometria armazenado em uma tabela em uma sequência de texto GML. O exemplo a seguir seleciona os valores que foram inseridos anteriormente na tabela SAMPLE_GEOMETRY. Os resultados mostrados no exemplo foram reformatados para facilitar a leitura. O espaçamento nos resultados pode variar de acordo com a exibição on-line.

```
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>
```

Alternativamente, você pode utilizar o grupo de transformação ST_GML para converter geometrias implicitamente em sua representação de HTML ao ligá-las.

```
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>
```

Não são utilizadas funções espaciais na instrução SELECT para converter a geometria.

Função que Cria Geometrias a Partir de Coordenadas

A função ST_Point cria geometrias não apenas a partir de formatos de troca de dados, mas também de valores de coordenadas numéricas. Este recurso é muito útil se os dados do local já estiverem armazenados no banco de dados.

Exemplos de Chamada de Funções Construtoras

Revise os exemplos de código para chamar funções construtoras, criar tabelas para conter a saída de funções construtoras e recuperar a saída para saber como usar as funções construtoras.

O exemplo a seguir insere uma linha na tabela SAMPLE_GEOMETRY com ID 100 e um valor de ponto com uma coordenada X de 30, uma coordenada Y de 40 e no sistema de referência espacial 1 utilizando a representação de coordenadas e a representação de WKT (Well-known Text). Em seguida, insere outra linha com ID 200 e um valor de sequência de linhas com as 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"
```

Se você souber que a coluna espacial pode conter apenas valores ST_Point, pode utilizar o exemplo a seguir, que insere dois pontos. A tentativa de inserir uma sequência de linhas ou qualquer outro tipo que não seja um ponto resultará em um erro de SQL. A primeira inserção cria uma geometria de ponto a partir da representação de WKT (Well-Know Text). A segunda cria uma geometria de ponto a partir de valores de coordenadas numéricas. Observe que esses valores de entrada também poderiam ser selecionados a partir de colunas de tabelas 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"
```

O exemplo a seguir utiliza SQL incorporada e assume que o aplicativo preenche as áreas de dados com os valores apropriados.

```
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;

// * A lógica do aplicativo para ler os buffers vai aqui */

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));
```

A amostra de código Java™ a seguir utiliza JDBC para inserir geometrias de pontos utilizando os valores de coordenadas numéricas X, Y e utiliza a representação de WKT para especificar as geometrias.

```
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);                    // valor do id
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);         // valor do id
pstmt.setDouble(2, 40.3);     // lat
pstmt.setDouble(3, -72.5);    // long
rc = pstmt.executeUpdate();
```

Funções de Comparação para Recursos Geográficos

Determinadas funções espaciais retornam informações sobre as formas com que os recursos geográficos relacionam-se ou comparam-se entre si. Outras funções espaciais retornam informações sobre se duas definições de sistemas de coordenadas ou dois sistemas de referência espacial são os mesmos.

Em todos os casos, as informações retornadas são resultados de uma comparação entre geometrias, entre definições de sistemas de coordenadas ou entre sistemas de referência espacial. As funções que fornecem estas informações são chamadas de funções de comparação. A tabela a seguir lista as funções de comparação por propósito.

Tabela 25. Funções de Comparação por Propósito

Propósito	Funções
Determina se o interior de uma geometria cruza o interior de outra.	• ST_Contains • ST_Within

Tabela 25. Funções de Comparação por Propósito (continuação)

Propósito	Funções
Retorna informações sobre interseções de geometrias.	• ST_Crosses • ST_Intersects • ST_Overlaps • ST_Touches
Determinam se o menor retângulo que inclui uma geometria cruza o menor retângulo que inclui outra geometria.	• ST_EnvIntersects • ST_MBRIntersects
Determinam se dois objetos são idênticos.	• ST_Equals • ST_EqualCoordsys • ST_EqualSRS
Determinam se as geometrias sendo comparadas atendem as condições da sequência de matrizes padrão DE-9IM.	• ST_Relate
Verifica se existe uma interseção entre duas geometrias.	• ST_Disjoint

No DB2 Spatial Extender, as funções de comparação retornam um valor de 1 (um) se uma comparação atender a alguns critérios, um valor de 0 (zero) se uma comparação falhar ao atender aos critérios e um valor nulo se a comparação não puder ser executada.

As comparações não poderão ser executadas se a operação de comparação não tiver sido definida para os parâmetros de entrada, ou se um dos parâmetros for nulo. As comparações poderão ser executadas se as geometrias com tipos de dados ou dimensões diferentes forem designadas aos parâmetros.

O DE-9IM (Dimensionally Extended 9 Intersection Model) é uma abordagem matemática que define a relação espacial de par inteligente entre as geometrias de tipos e dimensões diferentes. Esse modelo expressa relações espaciais entre todos os tipos de geometrias como interseções de pares inteligentes de seu interior, limite e exterior, considerando-se a dimensão das interseções resultantes.

As geometrias especificadas *a* e *b*: I(*a*), B(*a*) e E(*a*) representam o interior, o limite e o exterior de *a*, respectivamente. E, I(b), B(b), e E(b) representam o interior, o limite e o exterior de *b*. As interseções de I(a), B(a) e E(a) com I(b), B(b) e E(b) produzem uma matriz 3 por 3. Cada interseção pode resultar em geometrias de dimensões diferentes. Por exemplo, a interseção dos limites de dois polígonos consiste em um ponto e uma sequência de linhas, em cujo caso a função dim retorna a dimensão máxima de 1.

A função dim retorna um valor -1, 0, 1 ou 2. O valor -1 corresponde ao conjunto nulo ou dim(null), que é retornado quando nenhuma interseção foi localizada.

Os resultados retornados pelas funções de comparação podem ser entendidos ou verificados pela comparação dos resultados retornados por uma função de comparação com uma matriz padrão que representa os valores aceitáveis para o DE-9IM.

A matriz padrão contém os valores aceitáveis para cada uma das células matrizes de interseção. Os possíveis valores padrão são:

T Deve existir uma interseção; dim = 0, 1 ou 2.

F Não deve existir uma interseção; dim = -1.

***** Não importa se existir uma interseção; dim = -1, 0, 1 ou 2.

0 Deve existir uma interseção e sua dimensão exata deve ser 0; dim = 0.

1 Deve existir uma interseção e sua dimensão máxima deve ser 1; dim = 1.

2 Deve existir uma interseção e sua dimensão máxima deve ser 2; dim = 2.

A função ST_Within retorna um valor de 1 quando os interiores das duas geometrias se cruzam e quando o interior ou limite de *a* não cruza com o exterior de *b.* As outras condições não importam.

Cada função tem, pelo menos, uma matriz padrão, mas algumas requerem mais de uma para descrever as relações de várias combinações de tipos de geometria.

O DE-9IM foi desenvolvido por Clementini e Felice, que ampliaram dimensionalmente o Modelo de Interseção 9 de Egenhofer e Herring. O DE-9IM é uma colaboração de quatro autores (Clementini, Eliseo, Di Felice e van Osstrom) que publicaram o modelo em "A Small Set of Formal Topological Relationships Suitable for End-User Interaction," D. Abel e B.C. Ooi (Ed.), *Advances in Spatial Database - Terceiro Simpósio Internacional. SSD '93*. LNCS 692. Pp. 277-295. O modelo 9 Intersection por M. J. Egenhofer e J. Herring (Springer-Verlag Singapore [1993]) foi publicado no "Categorizing binary topological relationships between regions, lines, and points in geographic databases," *Tech. Report, Department of Surveying Engineering*, University of Maine, Orono, ME 1991.

Funções que Determinam se uma Geometria Contém Outra

ST_Contains e ST_Within obtêm duas geometrias como entrada e determinam se o interior de uma cruza o interior da outra.

Em termos coloquiais, ST_Contains determina se a primeira geometria especificada para ela inclui a segunda geometria (se a primeira contém a segunda). ST_Within determina se a primeira geometria está completamente dentro da segunda (se a primeira está contida na segunda).

Funções que Determinam se as Geometrias se Cruzam

As funções ST_Intersects, ST_Crosses, ST_Overlaps, e ST_Touches determinam se há intersecções entre geometrias.

Estas funções diferem principalmente quanto ao escopo de intersecção para o qual fazem o teste:

- ST_Intersects testa para determinar se as duas geometrias especificadas para ela atendem a uma das quatro condições: a de interseção dos interiores das geometrias, a de interseção de seus limites, a do limite da interseção da primeira geometria com o interior da segunda ou a do interior de interseção da primeira geometria com o limite da segunda.
- ST_Crosses é utilizada para analisar a interseção de geometrias de dimensões diferentes, com uma exceção: também pode analisar a interseção de sequências de linhas. Em todos os casos, o próprio local de interseção é considerado uma geometria; e ST_Crosses requer que essa geometria tenha uma dimensão menor do que a maior das interseções de geometrias (ou, se ambas forem sequências de linhas, que o local de interseção tenha uma dimensão menor do que uma sequência de linhas). Por exemplo, as dimensões de uma sequência de linhas e um polígono são 1 e 2, respectivamente. Se essas duas geometrias se cruzarem, e

se o local de interseção for linear (o caminho da sequência de linhas junto ao polígono), esse próprio local poderá ser considerado uma sequência de linhas. E como a dimensão de uma sequência de linhas (1) é menor do que a de um polígono (2), ST_Crosses, depois de analisar a interseção, retornaria um valor 1.

- As geometrias especificadas para ST_Overlaps como entrada devem ter a mesma dimensão. ST_Overlaps requer que essas geometrias sobreponham-se parcialmente, formando uma nova geometria (a região de sobreposição) que tem a mesma dimensão delas.
- ST_Touches determina se os limites das duas geometrias se cruzam.

Funções que Comparam Envelopes de Geometrias

ST_EnvIntersects e ST_MBRIntersects são funções semelhantes que determinam se o menor retângulo que inclui uma geometria tem intersecção com o menor retângulo que inclui outra geometria. Tradicionalmente, esse retângulo é chamado de envelope.

Multipolígonos, polígonos, sequências de múltiplas linhas e sequências de linhas curvas tocam os lados de seus envelopes; sequências de linhas horizontais, sequências de linhas verticais e pontos são levemente menores que seus envelopes. ST_EnvIntersects testa para determinar se os envelopes de geometrias se cruzam.

A menor área retangular na qual uma geometria pode se ajustar é chamada de MBR (Minimum Bounding Rectangle). Os envelopes em volta dos multipolígonos, polígonos, sequências de múltiplas linhas e sequências de linhas curvas são realmente MBRs. Mas os envelopes em volta de sequências de linhas horizontais, sequências de linhas verticais e pontos não são MBRs, porque não constituem uma área mínima em que essas últimas geometrias se encaixam. Essas últimas geometrias não ocupam espaço definível e, portanto, não podem ter MBRs. Contudo, foi adotada uma convenção por meio da qual são referidas como seus próprios MBRs. Portanto, em relação aos multipolígonos, polígonos, sequências de múltiplas linhas e sequências de linhas curvas, ST_MBRIntersects testa a interseção dos mesmos retângulos circundantes que ST_EnvIntersects testa. Mas para sequências de linhas horizontais, sequências de linhas verticais e pontos, ST_MBRIntersects testa as interseções dessas próprias geometrias.

Funções que Verificam se Dois Itens São Idênticos

Estas funções comparam sistemas de referência espacial, definições de sistemas de coordenadas ou geometrias.

- ST_EqualCoordsys
- ST_Equals
- ST_EqualSRS

Função que Determina se as Geometrias não se Cruzam

ST_Disjoint retornará um valor 1 (um) se a interseção das duas geometrias for um conjunto vazio. Essa função retorna o oposto exato do que é retornado por ST_Intersects.

A ilustração mostra diferentes geometrias e como os limites não se cruzam em nenhum ponto.

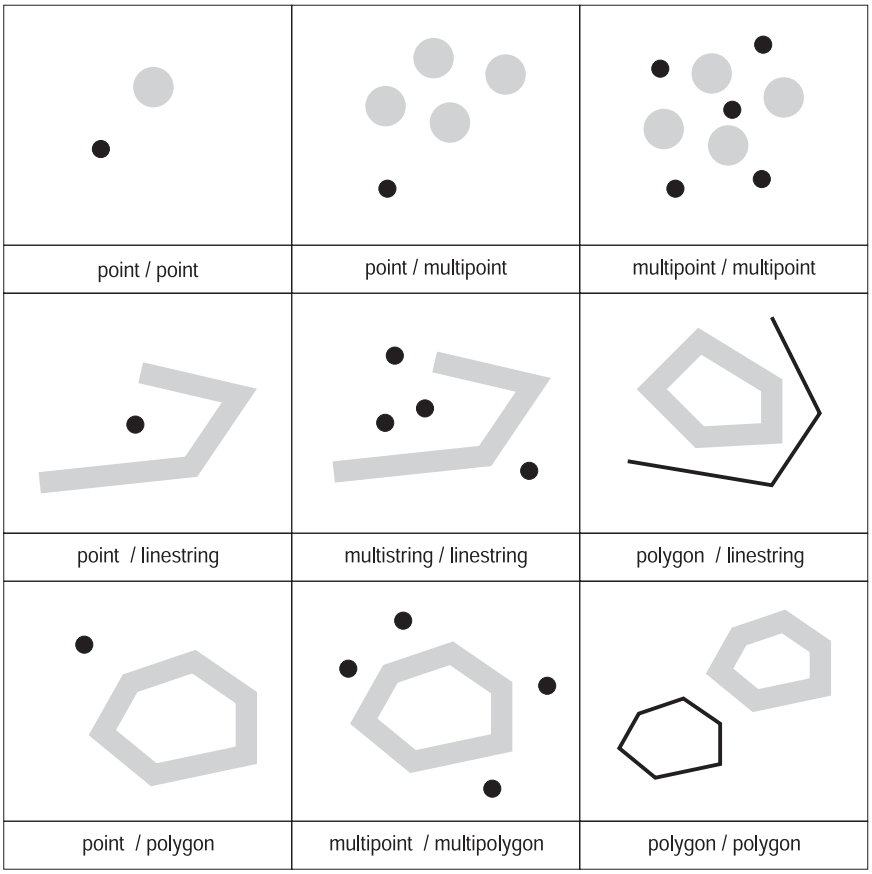


Figura 17. ST_Disjoint. As geometrias escuras representam a geometria a; as geometrias cinzas representam a geometria b. Em todos os casos, a geometria a e a geometria b são desconectadas uma da outra.

A matriz a seguir apenas indica que nem os interiores nem os limites de nenhuma das geometrias têm intersecção.

Tabela 26. Matriz para ST_Disjoint

	Interior da Geometria b	**Limite da Geometria b**	**Exterior da Geometria b**
Limite da Geometria a	F	F	*
Interior da Geometria a	F	F	*
Exterior da Geometria a	*	*	*

Função que Compara Duas Geometrias à Sequência de Matrizes Padrão DE-9IM

A função ST_Relate comparará duas geometrias e retornará um valor 1 (um) se as geometrias atenderem às condições especificadas pela sequência de matrizes do padrão DE-9IM; caso contrário, a função retornará um valor 0 (zero).

Funções para Obter Informações sobre Geometrias e Índices

O DB2 Spatial Extender fornece funções que retornam informações sobre propriedades de geometrias e índices espaciais.

As informações retornadas estão relacionadas a:

- Tipos de dados de geometrias
- Coordenadas e medidas em uma geometria
- Anéis, limites, envelopes e MBRs (retângulos de limite mínimo)
- Dimensões
- As qualidades de ser fechado, vazio ou simples
- Geometrias base em uma coleção de geometrias
- Sistemas de Referência Espacial
- Distância entre geometrias
- Parâmetros usados para definir um índice espacial ou um índice em uma coluna espacial

Algumas propriedades são geometrias em seu próprio direito; por exemplo, os anéis exteriores e interiores de uma superfície, ou os pontos iniciais e nós de extremidade de uma curva. Essas geometrias são produzidas por algumas das funções nessa categoria. Funções que produzem outros tipos de geometrias — por exemplo, geometrias que representam zonas que circundam um determinado local — pertencem à categoria "Funções espaciais que geram novas geometrias".

Função que Retorna Informações de Tipos de Dados

ST_GeometryType utiliza uma geometria como parâmetro de entrada e retorna o nome completo do tipo dinâmico dessa geometria.

Funções que Retornam Informações de Coordenadas e Medidas

As seguintes funções retornam informações sobre as coordenadas e medidas contidas em uma geometria. Por exemplo, ST_X pode retornar a coordenada X contida em um ponto especificado, ST_MaxX retorna a maior coordenada X contida em uma geometria e ST_MinX retorna a menor coordenada X contida em uma geometria.

Estas funções são:

- 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

Funções que Retornam Informações sobre Geometrias em uma Geometria

As seguintes funções retornam informações sobre geometrias contidas em uma geometria. Algumas funções identificam pontos específicos contidos em uma geometria; outras retornam o número de geometrias base contidas em uma coleção.

Estas funções são:

- ST_Centroid
- ST_EndPoint
- ST_GeometryN
- ST_LineStringN
- ST_MidPoint
- ST_NumGeometries
- ST_NumLineStrings
- ST_NumPoints
- ST_NumPolygons
- ST_PointN
- ST_PolygonN
- ST_StartPoint

Funções que Retornam Informações sobre Limites, Envelopes e Anéis

As funções a seguir retornam informações sobre demarcações que dividem uma parte interna de uma geometria a partir de uma parte externa, ou que divide a própria geometria a partir do espaço externo a ela. Por exemplo, ST_Boundary retorna o limite de uma geometria na forma de uma curva.

Estas funções são:

- ST_Boundary
- ST_Envelope
- ST_EnvIntersects
- ST_ExteriorRing
- ST_InteriorRingN
- ST_MBR
- ST_MBRIntersects
- ST_NumInteriorRing
- ST_Perimeter

Funções que Retornam Informações sobre Dimensões de uma Geometria

As funções a seguir retornam informações sobre a dimensão de uma geometria, tal como, a área de cobertura de uma determinada geometria ou o comprimento de uma determinada curva ou multicurva.

Estas funções são:

- ST_Area
- ST_Dimension
- ST_Length

Funções que Indicam se uma Geometria É Fechada, Vazia ou Simples

O DB2 Spatial Extender fornece funções que indicam se uma geometria é fechada, vazia ou simples.

Estas funções indicam:

- Se uma determinada curva ou multicurva é fechada (isto é, se o ponto inicial e final da curva ou multicurva são os mesmos)
- Se uma determinada geometria é vazia (isto é, sem pontos)
- Se uma curva, multicurva ou multiponto é simples (isto é, se essas geometrias possuem configurações típicas)

Estas funções são:

- ST_IsClosed
- ST_IsEmpty
- ST_IsSimple

Funções que Identificam o Sistema de Referência Espacial Associado a uma Geometria

As seguintes funções retornam valores que identificam o sistema de referência espacial que foi associado à geometria. Além disso, a função ST_SrsID pode alterar o sistema de referência espacial da geometria sem alterar ou transformar a geometria.

Estas funções são:

- ST_SrsId (Também Chamada ST_SRID)
- ST_SrsName

Função que Retorna Informações de Distância entre Geometrias

ST_Distance utiliza duas geometrias e, opcionalmente, uma unidade como parâmetros de entrada e retorna a distância mais curta entre qualquer ponto na primeira geometria para qualquer ponto na segunda geometria, medido nas unidades especificadas.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria, ela será convertida para o outro sistema de referência espacial.

Se uma das duas geometrias for um valor nulo ou for vazia, será retornado um valor nulo.

Por exemplo, ST_Distance poderá informar a distância mais curta que uma aeronave deve viajar entre dois locais. A Figura 18 na página 245 ilustra essas informação.

A figura mostra um mapa dos Estados Unidos com uma linha reta entre os pontos rotulados Los Angeles e Chicago.

Figura 18. Distância mínima entre duas cidades. ST_Distance pode ter as coordenadas para os locais de Los Angeles e Chicago como entrada e retornar um valor indicando a distância mínima entre esses locais.

Função que Retorna Informações de Índice

ST_GetIndexParms utiliza o identificador para um índice espacial ou para uma coluna espacial como parâmetro de entrada e retorna os parâmetros utilizados para definir o índice ou o índice na coluna espacial.

Se um número de parâmetros adicionais for especificado, apenas o parâmetro identificado pelo número será retornado.

Funções para Gerar Novas Geometrias a partir de Geometrias Existentes

Esta seção apresenta a categoria de funções que originam novas geometrias de geometrias existentes.

Essa categoria não inclui funções que originam geometrias que representam propriedades de outras geometrias. Em vez disso, serve para funções que:

- Convertem geometrias em outras geometrias
- Criam geometrias que representam configurações de espaço
- Originam geometrias individuais de várias geometrias
- Criam geometrias com base em medidas
- Criam modificações de geometrias

Funções que Convertem uma Geometria de um Supertipo em um Subtipo Correspondente

As funções a seguir podem converter geometrias de um supertipo em geometrias correspondentes de um subtipo.

Por exemplo, a função ST_ToLineString pode converter uma sequência de linhas do tipo ST_Geometry em uma sequência de linhas de ST_LineString. Algumas dessas funções também podem combinar geometrias base e coleções de geometrias em uma única coleção de geometrias. Por exemplo, ST_ToMultiLine pode converter uma cadeia de linhas e uma cadeia de múltiplas linhas em uma única cadeia de múltiplas linhas.

- ST_Polygon
- ST_ToGeomColl
- ST_ToLineString

- ST_ToMultiLine
- ST_ToMultiPoint
- ST_ToMultiPolygon
- ST_ToPoint
- ST_ToPolygon

Funções que Criam Novas Geometrias com Diferentes Configurações de Espaço

Utilizando geometrias existentes como ponto inicial, as funções a seguir criam novas geometrias que representam áreas circulares ou outras configurações de espaço. Por exemplo, supondo um ponto que representa o centro de um aeroporto indicado, ST_Buffer pode criar uma superfície que representa, na forma circular, a extensão proposta do aeroporto.

Estas funções são:

- ST_Buffer
- ST_ConvexHull
- ST_Difference
- ST_Intersection
- ST_SymDifference

Funções que Derivam uma Nova Geometria a partir de Diversas

As funções a seguir originam geometrias individuais de várias geometrias. Por exemplo, combinando duas geometrias em uma única geometria.

- MBR Aggregate
- ST_Union
- Union Aggregate

Funções que Criam Nova Geometria Baseada em Medidas de Geometria Existentes

Estas funções podem criar uma nova geometria baseada em medidas a partir de uma geometria existente. Elas também podem retornar a distância para um local ao longo da geometria.

Estas funções são:

- ST_DistanceToPoint
- ST_FindMeasure ou ST_LocateAlong
- ST_MeasureBetween ou ST_LocateBetween
- ST_PointAtDistance

Funções que Criam Formas Modificadas de Geometrias Existentes

As funções a seguir criam formas modificadas de geometrias existentes, como por exemplo, a criação de versões estendidas de curvas existentes. Cada versão inclui os pontos em uma curva existente mais um ponto adicional.

Estas funções são:

- ST_AppendPoint
- ST_ChangePoint
- ST_Generalize
- ST_M

- ST_PerpPoints
- ST_RemovePoint
- ST_X
- ST_Y
- ST_Z

Funções que Convertem Geometrias entre Sistemas de Coordenadas

ST_Transform utiliza uma geometria e um identificador do sistema de referência espacial como parâmetros de entrada e transforma a geometria a ser representada no sistema de referência espacial especificado.

As projeções e conversões entre diferentes sistemas de coordenadas são executadas e as coordenadas das geometrias são ajustadas de acordo.

Função EnvelopesIntersect

Use a função EnvelopesIntersect para determinar se duas geometrias se cruzam ou uma geometria se cruza com um envelope definido pelos quatro valores de tipo DOUBLE.

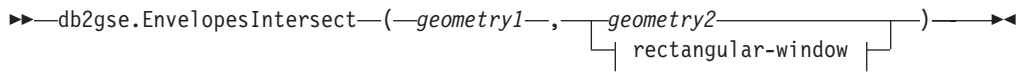
EnvelopesIntersect aceita dois tipos de parâmetros de entrada:

- Duas geometrias

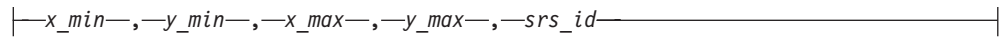
  EnvelopesIntersect retornará 1 se o envelope da primeira geometria cruzar com o envelope da segunda. Caso contrário, será retornado 0 (zero).
- Uma geometria, quatro valores de coordenadas do tipo DOUBLE que definem os cantos inferior esquerdo e superior direito de uma janela retangular e o identificador do sistema de referência espacial.

  EnvelopesIntersect retorna 1 se o envelope da primeira geometria fizer interseção com o envelope definido pelos quatro valores do tipo DOUBLE. Caso contrário, será retornado 0 (zero).

Sintaxe



janela retangular:



Parâmetros

geometry1
Um valor do tipo ST_Geometry ou um dos seus subtipos que representam a geometria cujo envelope será testado quanto à interseção com o envelope de *geometry2* ou com a janela retangular definida pelos quatro valores do tipo DOUBLE.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria cujo envelope será testado quanto à interseção com o envelope de *geometry1*.

x_min
Especifica o valor mínimo da coordenada X para o envelope. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é DOUBLE.

y_min
Especifica o valor mínimo da coordenada Y para o envelope. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é DOUBLE.

x_max
Especifica o valor máximo da coordenada X para o envelope. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é DOUBLE.

y_max
Especifica o valor máximo da coordenada Y para o envelope. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é DOUBLE.

srs_id
Identifica exclusivamente o sistema de referência espacial. O identificador do sistema de referência espacial deve corresponder ao identificador do sistema de referência espacial do parâmetro de geometria. Você deve especificar um valor diferente de nulo para este parâmetro.

O tipo de dados deste parâmetro é INTEGER.

Tipo de retorno

INTEGER

Exemplo

Este exemplo cria dois polígonos que representam regiões e depois determina se uma delas faz interseção com uma área geográfica especificada pelos quatro valores do tipo DOUBLE.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE counties (id INTEGER, name CHAR(20), geometry ST_Polygon)

INSERT INTO counties VALUES
       (1, 'County_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 counties VALUES
       (2, 'County_2', ST_Polygon('polygon((15 15, 15 20, 60 20, 60 15,
        15 15))' ,0))

INSERT INTO counties VALUES
       (3, 'County_3', ST_Polygon('polygon((115 15, 115 20, 160 20, 160 15,
        115 15))' ,0))


SELECT name
FROM counties as c
WHERE EnvelopesIntersect(c.geometry, 15, 15, 60, 20, 0) =1
```

Resultados:

```
Nome
--------------------
County_1
County_2
```

Funções Agregadas MBR

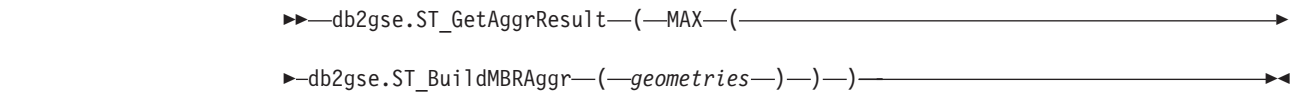
Use as funções ST_BuildMBRAggr e ST_GetAggrResult combinadas para agregar um conjunto de geometrias em uma coluna a uma única geometria. A combinação constrói um retângulo que representa o minimum bounding rectangle que inclui todas as geometrias na coluna. As coordenadas Z e M são descartadas quando o agregado é calculado.

A expressão a seguir é um exemplo que utiliza a função MAX com a função espacial db2gse.ST_BuildMBRAggr para calcular o MBR das geometrias na coluna columnName e a função espacial db2gse.ST_GetAggrResult para retornar a geometria resultante que foi calculada para o MBR:

```
db2gse.ST_Get_AggrResult(MAX(db2gse.ST_BuildMBRAggr(columnName)))
```

Se todas as geometrias a serem combinadas forem nulas, será retornado nulo. Se todas as geometrias forem nulas ou vazias, será retornada uma geometria vazia. Se o retângulo limite mínimo de todas as geometrias a serem combinadas resultar em um ponto, este ponto será retornado como valor ST_Point. Se o retângulo limite mínimo de todas as geometrias a serem combinadas resultar em uma sequência de linha horizontal ou vertical, essa sequência de linha será retornada como um valor ST_LineString. Caso contrário, o retângulo limite mínimo será retornado como um valor ST_Polygon.

Sintaxe



Parâmetro

MVS/ESA

Uma coluna selecionada do tipo ST_Geometry ou um dos seus subtipos e que representa todas as geometrias para as quais o retângulo limite mínimo deve ser calculado.

Tipo de retorno

db2gse.ST_Geometry

Restrições

Não é possível construir o agregado de união de uma coluna espacial em uma seleção completa em qualquer uma das seguintes situações:

- Em um ambiente de banco de dados particionado
- Se a cláusula GROUP BY for utilizada na seleção completa
- Se você utilizar uma função diferente da função agregada do DB2 MAX.

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo mostra como utilizar a função ST_BuildMBRAggr para obter o retângulo limite máximo de todas as geometrias dentro de uma coluna. Neste exemplo, diversos pontos são incluídos na coluna GEOMETRY na tabela SAMPLE_POINTS. Em seguida, o código SQL determina o retângulo limite máximo de todos os pontos 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))
```

Função ST_AppendPoint

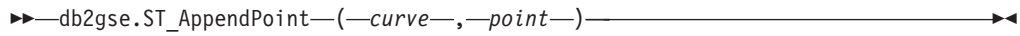
A função ST_AppendPoint usa uma curva e um ponto como parâmetros de entrada e estende a curva pelo ponto especificado. Se a curva especificada tiver coordenadas Z ou M, o ponto também deverá ter coordenadas Z ou M. A curva resultante é representada no sistema de referência espacial da curva especificada.

Se o ponto a ser anexado não for representado no mesmo sistema de referência espacial que a curva, ele será convertido para o outro sistema de referência espacial.

Se a curva especificada for fechada ou simples, a curva resultante pode não ser mais fechada ou simples. Se a curva ou ponto especificado for nulo, ou se a curva for vazia, será retornado nulo. Se o ponto a ser anexado for vazio, a curva especificada será retornada inalterada e um aviso será retornado (SQLSTATE 01HS3).

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

curve Um valor do tipo ST_Curve ou um de seus subtipos que representa a curva à qual *point* será anexado.

ponto Um valor do tipo ST_Point que representa o ponto que está anexado a *curve*.

Tipo de retorno

db2gse.ST_Curve

Exemplos

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este código cria duas sequências de linhas, cada uma com três pontos.

```
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) )
```

Exemplo 1

Este exemplo inclui o ponto (5, 5) no final de uma sequência de linhas.

```
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)
```

Exemplo 2

Este exemplo inclui o ponto (15, 15, 7) no final de uma sequência de linhas com 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)
```

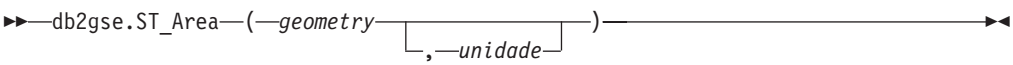
Função ST_Area

A função ST_Area usa uma geometria e, opcionalmente, uma unidade como parâmetros de entrada e retorna a área coberta pela geometria na unidade de medida padrão ou especificada.

Se a geometria for um polígono ou multipolígono, a área coberta pela geometria será retornada. A área de pontos, cadeias de linhas, multipontos e cadeias multilinha é 0 (zero). Se a geometria for nula ou vazia, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que determina a área.

unit Um valor VARCHAR(128) que identifica as unidades nas quais a área é medida. As unidades de medida suportadas estão listadas na exibição do catálogo DB2GSE.ST_UNITS_OF_MEASURE.

Se o parâmetro *unit* for omitido, as seguintes regras serão utilizadas para determinar a unidade na qual a área será medida:

- Se *geometry* estiver em um sistema de coordenadas projetadas ou geocêntricas, a unidade linear associada a este sistema de coordenadas será utilizada.
- Se *geometry* estiver em um sistema de coordenadas geográficas, a unidade angular associada a este sistema de coordenadas será usada.

Restrições para conversões de unidades: Será retornado um erro (SQLSTATE 38SU4) se ocorrer qualquer uma das condições a seguir:

- A geometria está em um sistema de coordenadas não especificado e o parâmetro *unit* é especificado.
- A geometria está em um sistema de coordenadas projetadas e uma unidade angular é especificada.
- A geometria está em um sistema de coordenadas geográficas e uma unidade linear é especificada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

O analista espacial precisa de uma lista da área coberta por cada região de vendas. Os polígonos da região de vendas são armazenados na tabela SAMPLE_POLYGONS. A área é calculada pela aplicação da função ST_Area na coluna da geometria.

```
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))
```

A instrução SELECT a seguir recupera o ID e a área da região de vendas:

```
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
```

Exemplo 2

A instrução SELECT a seguir recupera o ID e a área da região de vendas em várias 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
```

Exemplo 3

Este exemplo encontra a área de um polígono definida nas coordenadas de Plano de Estado.

O sistema de referência espacial Plano de Estado com um ID 3 é criado com o seguinte comando:

```
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
```

As seguintes instruções SQL adicionam o polígono, no sistema de referência espacial 3, à tabela e determinam a área em pés quadrados, em metros quadrados e em milhas quadradas.

```
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  SQUARE FEET           SQUARE METERS          SQUARE MILES
-- --------------------- --------------------- ---------------------
 1 +7.94698788000000E+008   +7.38302286101346E+007   +2.85060106320552E+001
```

Função ST_AsBinary

A função ST_AsBinary usa uma geometria como um parâmetro de entrada e retorna sua representação binária reconhecida. As coordenadas Z e M são descartadas e não farão parte da representação binária reconhecida.

Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos a serem convertidos para a representação binária reconhecida correspondente.

Tipo de retorno

BLOB(2G)

Exemplos

O código a seguir ilustra como utilizar a função ST_AsBinary para converter os pontos nas colunas da geometria da tabela SAMPLE_POINTS em representação binária reconhecida (WKB) na coluna 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))
```

Exemplo 1

Este exemplo ocupa a coluna WKB com um ID 1111, a partir da coluna GEOMETRY, com um ID 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)
```

Exemplo 2

Este exemplo exibe a representação binária 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'
```

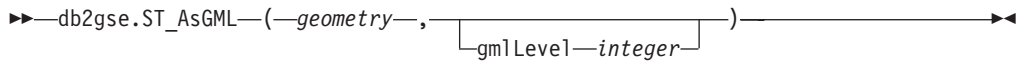
Função ST_AsGML

A função ST_AsGML usa uma geometria como um parâmetro de entrada e retorna sua representação usando geography markup language.

Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

gmlLevel
Um parâmetro opcional que especifica o nível de especificação GML a ser utilizado para formatar os dados GML a serem retornados. Os valores válidos são:

- 2 - Utilize o nível de especificação GML 2 com a tag <gml:coordinates>.
- 3 - Utilize o nível de especificação GML 3 com a tag <gml:poslist>.

Se nenhum parâmetro for especificado, a saída é retornada utilizando a especificação GML de nível 2 com a tag <gml:coord>.

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos a serem convertidos na representação GML correspondente.

Tipo de retorno

CLOB(2G)

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

O fragmento de código a seguir ilustra como utilizar a função ST_AsGML para exibir o fragmento de GML. Este exemplo ocupa a coluna GML, a partir da coluna da geometria, com um ID 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))
```

A instrução SELECT a seguir lista o ID e as representações GML da geometria.

```
SELECT id, cast(ST_AsGML(geometry) AS varchar(110)) AS gml_fragment
FROM sample_points
WHERE  id = 1100
```

Resultados:

```
SELECT id,
       cast(ST_AsGML(geometry) AS varchar(110)) AS gml,
       cast(ST_AsGML(geometry,2) AS varchar(110)) AS gml2,
       cast(ST_AsGML(geometry,3) AS varchar(110)) AS gml3
FROM sample_points
WHERE  id = 1100
```

A instrução SELECT retorna o seguinte conjunto de resultados:

```
ID          GML                                 GML2                                GML3
----------- ----------------------------------  ----------------------------------  ----------------------------------
       1100 <gml:Point srsName="EPSG:4269">     <gml:Point srsName="EPSG:4269">     <gml:Point srsName="EPSG:4269
                                                                                    srsDimension="2">
            <gml:coord>                         <gml:coordinates>                   <gml:pos>
            <gml:X>10</gml:X><gml:Y>20</gml:Y> 10,20                               10,20
            </gml:coord></gml:Point>            </gml:coordinates></gml:Point>      </gml:pos></gml:Point>
```

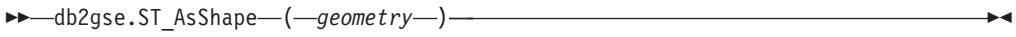
Função ST_AsShape

A função St_AsShape usa uma geometria como um parâmetro de entrada e retorna sua representação de forma ESRI.

Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos a serem convertidos para a representação da forma ESRI correspondente.

Tipo de retorno

BLOB(2G)

Exemplo

O fragmento de código a seguir ilustra como utilizar a função ST_AsShape para converter os pontos na coluna da geometria da tabela SAMPLE_POINTS em representação binária de formatos na coluna BLOB de formato. Este exemplo ocupa a coluna de formato a partir da coluna da geometria. A representação binária de formatos é utilizada para exibir as geometrias em geonavegadores, que requerem que as geometrias estejam em conformidade com o formato do arquivo modelo ESRI, ou que as geometrias sejam construídas para o arquivo *.SHP do arquivo modelo.

```
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
```

Retorna:

```
ID     SHAPE

------ --------------------------------------------
  1100 x'0100000000000000000024400000000000003440'
```

Função ST_AsText

A função ST_AsText usa uma geometria como um parâmetro de entrada e retorna sua representação de texto reconhecida.

Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos a serem convertidos para a representação de texto reconhecida correspondente.

Tipo de retorno

CLOB(2G)

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Depois de capturar e inserir dados na tabela SAMPLE_GEOMETRIES, um analista deseja verificar se os valores inseridos estão corretos, examinando a representação de texto reconhecida das geometrias.

```
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))
```

A instrução SELECT a seguir lista o tipo espacial e a representação WKT das geometrias. A geometria é convertida em texto pela função ST_AsText. Então é feita uma conversão em um varchar(120) porque a saída padrão da função ST_AsText é 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))
```

Função ST_Boundary

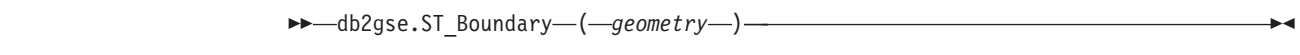
A função ST_Boundary usa uma geometria como um parâmetro de entrada e retorna seu limite como uma nova geometria. A geometria resultante é representada no sistema de referência espacial da geometria especificada.

Se a geometria especificada for um ponto, multiponto, curva fechada ou multicurva fechada, ou se for fazia, o resultado será uma geometria vazia do tipo ST_Point. Para curvas ou multicurvas que não são fechadas, os pontos iniciais e finais das curvas são retornados como um valor ST_MultiPoint, a menos que esse ponto seja o ponto inicial ou o final de um número de curvas par. Para superfícies e multisuperfícies, a curva que define o limite da geometria especificada é retornada como um valor ST_Curve ou ST_MultiCurve. Se a geometria indicada for nula, será retornado nulo.

Se possível, o tipo específico da geometria retornada será ST_Point, ST_LineString ou ST_Polygon. Por exemplo, o limite de um polígono sem aberturas é uma sequência de linhas única, representada como ST_LineString. O limite de um polígono com uma ou mais aberturas consiste em várias sequências de linhas, representadas como ST_MultiLineString.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos. O limite desta geometria é retornado.

Tipo de retorno

db2gse.ST_Geometry

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo cria várias geometrias e determina o limite de cada uma delas.

```
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
```

Função ST_Buffer

A função ST_Buffer usa uma geometria, uma distância e, opcionalmente, uma unidade como parâmetros de entrada e retorna a geometria que circunda a geometria especificada pela distância especificada, medida na unidade especificada.

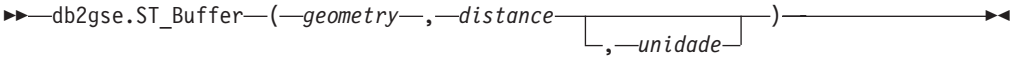
Cada ponto no limite da geometria resultante representa a distância especificada da geometria especificada. A geometria resultante é representada no sistema de referência espacial da geometria especificada.

Qualquer curva circular no limite da geometria resultante é aproximada por sequências lineares. Por exemplo, o buffer em torno de um ponto, que pode resultar em uma região circular, é aproximado por um polígono cujo limite é uma sequência de linhas.

Se a geometria especificada for nula ou estiver vazia, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria para criar o buffer ao redor.

distance
Um valor DOUBLE PRECISION que especifica a distância a ser usada para o buffer em torno de *geometry*.

unit Um valor VARCHAR(128) que identifica a unidade na qual a *distância* é calculada. As unidades de medida suportadas estão listadas na exibição do catálogo DB2GSE.ST_UNITS_OF_MEASURE.

Se o parâmetro *unit* for omitido, as seguintes regras serão utilizadas para determinar a unidade de medida utilizada para a distância:

- Se *geometry* estiver em um sistema de coordenadas projetadas ou geocêntricas, a unidade linear associada a este sistema de coordenadas será a padrão.
- Se *geometry* estiver em um sistema de coordenadas geográficas, a unidade angular associada a este sistema de coordenadas será o padrão.

Restrições em conversões de unidades: Um erro (SQLSTATE 38SU4) será retornado se ocorrer qualquer uma das condições a seguir:

- A geometria está em um sistema de coordenadas não especificado e o parâmetro *unit* é especificado.
- A geometria está em um sistema de coordenadas projetadas e uma unidade angular é especificada.
- A geometria está em um sistema de coordenadas geográficas, mas não é um valor ST_Point e uma unidade linear é especificada.

Tipo de retorno

db2gse.ST_Geometry

Exemplos

Nos exemplos a seguir, os resultados foram reformatados para possibilitar a leitura. O espaçamento nos resultados irá variar de acordo com a exibição.

Exemplo 1

O código a seguir cria um sistema de referência espacial, cria a tabela SAMPLE_GEOMETRIES e ocupa-a.

```
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))
```

Exemplo 2

A instrução SELECT a seguir utiliza a função ST_Buffer para aplicar um buffer de 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))
```

Exemplo 3

A instrução SELECT a seguir utiliza a função ST_Buffer para aplicar um buffer negativo de 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))
```

Exemplo 4

A instrução SELECT a seguir mostra o resultado da aplicação de um buffer com o parâmetro unit 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))
```

Funções Agregadas MBR

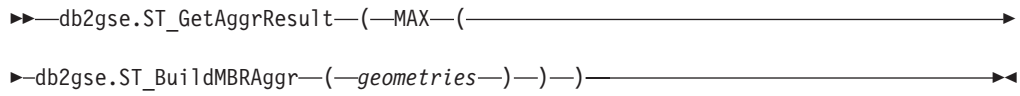
Use as funções ST_BuildMBRAggr e ST_GetAggrResult combinadas para agregar um conjunto de geometrias em uma coluna a uma única geometria. A combinação constrói um retângulo que representa o minimum bounding rectangle que inclui todas as geometrias na coluna. As coordenadas Z e M são descartadas quando o agregado é calculado.

A expressão a seguir é um exemplo que utiliza a função MAX com a função espacial db2gse.ST_BuildMBRAggr para calcular o MBR das geometrias na coluna columnName e a função espacial db2gse.ST_GetAggrResult para retornar a geometria resultante que foi calculada para o MBR:

```
db2gse.ST_Get_AggrResult(MAX(db2gse.ST_BuildMBRAggr(columnName)))
```

Se todas as geometrias a serem combinadas forem nulas, será retornado nulo. Se todas as geometrias forem nulas ou vazias, será retornada uma geometria vazia. Se o retângulo limite mínimo de todas as geometrias a serem combinadas resultar em um ponto, este ponto será retornado como valor ST_Point. Se o retângulo limite mínimo de todas as geometrias a serem combinadas resultar em uma sequência de linha horizontal ou vertical, essa sequência de linha será retornada como um valor ST_LineString. Caso contrário, o retângulo limite mínimo será retornado como um valor ST_Polygon.

Sintaxe



Parâmetro

MVS/ESA

Uma coluna selecionada do tipo ST_Geometry ou um dos seus subtipos e que representa todas as geometrias para as quais o retângulo limite mínimo deve ser calculado.

Tipo de retorno

db2gse.ST_Geometry

Restrições

Não é possível construir o agregado de união de uma coluna espacial em uma seleção completa em qualquer uma das seguintes situações:

- Em um ambiente de banco de dados particionado
- Se a cláusula GROUP BY for utilizada na seleção completa
- Se você utilizar uma função diferente da função agregada do DB2 MAX.

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo mostra como utilizar a função ST_BuildMBRAggr para obter o retângulo limite máximo de todas as geometrias dentro de uma coluna. Neste exemplo, diversos pontos são incluídos na coluna GEOMETRY na tabela SAMPLE_POINTS. Em seguida, o código SQL determina o retângulo limite máximo de todos os pontos 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))
```

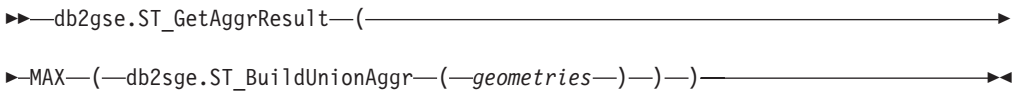
Funções Agregadas de União

Um agregado de união é a combinação das funções ST_BuildUnionAggr e ST_GetAggrResult. Use esta combinação para agregar uma coluna de geometrias em uma tabela a uma única geometria, construindo a união.

Se todas as geometrias a serem combinadas na união forem nulas, será retornado nulo. Se cada uma das geometrias a serem combinadas na união forem nulas ou vazias, será retornada uma geometria vazia do tipo ST_Point.

A função ST_BuildUnionAggr também pode ser chamada como um método.

Sintaxe



Parâmetros

MVS/ESA

Uma coluna em uma tabela do tipo ST_Geometry ou um dos seus subtipos e representa todas as geometrias que devem ser combinadas em uma união.

Tipo de retorno

db2gse.ST_Geometry

Restrições

Não é possível construir o agregado de união de uma coluna espacial em uma tabela em qualquer uma das seguintes situações:

- Em ambientes de banco de dados particionado
- Se uma cláusula GROUP BY for utilizada na seleção
- Se você utilizar uma função diferente da função agregada do DB2 MAX

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como um agregado de união pode ser utilizado para combinar um conjunto de pontos em multipontos. Diversos pontos são incluídos na tabela SAMPLE_POINTS. As funções ST_GetAggrResult e ST_BuildUnionAggr são utilizadas para construir a união dos pontos.

```
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)
```

Função ST_Centroid

A função ST_Centroid usa uma geometria como um parâmetro de entrada e retorna o centro geométrico, que é o centro do minimum bounding rectangle da geometria especificada, como um ponto. O ponto resultante é representado no sistema de referência espacial da geometria especificada.

Se a geometria especificada for nula ou vazia, então nulo é retornado.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria para determinar o centro geométrico.

Tipo de retorno

db2gse.ST_Point

Exemplo

Este exemplo cria duas geometrias e encontra o centróide delas.

```
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)
```

Função ST_ChangePoint

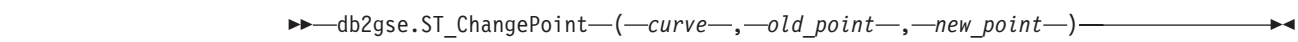
A função ST_ChangePoint usa uma curva e dois pontos como parâmetros de entrada. Substitui todas as ocorrências do primeiro ponto na curva especificada pelo segundo ponto e retorna a curva resultante. A geometria resultante é representada no sistema de referência espacial da geometria especificada.

Se os dois pontos não forem representados no mesmo sistema de referência espacial como a curva, eles serão convertidos para o sistema de referência espacial utilizado para a curva.

Se a curva for vazia, então um valor vazio será retornado. Se a curva especificada for nula ou se qualquer um dos pontos fornecidos for nulo ou vazio, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

curve Um valor do tipo ST_Curve ou um de seus subtipos que representa a curva em que os pontos identificados por *old_point* são alterados para *new_point*.

old_point
Um valor do tipo ST_Point que identifica os pontos na curva que são alterados para *new_point*.

new_point
Um valor do tipo ST_Point que representa as novas localizações dos pontos na curva identificada por *old_point*.

Tipo de retorno

db2gse.ST_Curve

Restrições

O ponto a ser alterado na curva deve ser um dos pontos utilizados para definir a curva.

Se a curva tiver coordenadas Z ou M, os pontos especificados também deverão ter coordenadas Z ou M.

Exemplos

Exemplo 1

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

O código a seguir cria e ocupa a tabela 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) )
```

Exemplo 2

Este exemplo altera todas as ocorrências do ponto (5, 5) para o ponto (6, 6) na sequência de linhas.

```
SELECT cast(ST_AsText(ST_ChangePoint(line, ST_Point(5, 5),
                                     ST_Point(6, 6))) as VARCHAR(160))
FROM sample_lines
WHERE  id=1
```

Exemplo 3

Este exemplo altera todas as ocorrências do ponto (5, 5, 5) para o ponto (6, 6, 6) na sequência de linhas.

```
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
```

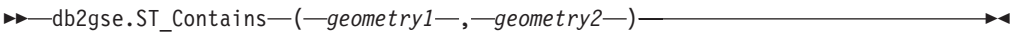
Resultados:

```
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)
```

Função ST_Contains

Use a função ST_Contains para determinar se uma geometria é totalmente contida por outra geometria.

Sintaxe



Parâmetro

geometry1
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que deve ser testada para conter totalmente *geometry2*.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que deve ser testada para que esteja totalmente dentro de *geometry1*.

Tipo de retorno

INTEGER

Uso

ST_Contains usa duas geometrias como parâmetro de entrada e retorna 1 se a primeira geometria contiver totalmente a segunda ou a segunda geometria for totalmente contida pela primeira geometria. Caso contrário, retorna 0 (zero) para indicar que a primeira geometria não contém totalmente a segunda.

A função ST_Contains retorna o resultado oposto exato da função ST_Within.

Se qualquer uma das geometrias especificadas for nula ou vazia, será retornado nulo.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria e usar o mesmo datum subjacente, ela será convertida no outro sistema de referência espacial.

A matriz de padrão da função ST_Contains indica que os interiores das duas geometrias devem se cruzar e que o interior ou limite da segunda (geometria *b*)

não deve cruzar o exterior da primeira (geometria *a*). O asterisco (*) indica que não importa se existe uma interseção entre essas partes das geometrias.

Tabela 27. Matriz para ST_Contains

	Interior da Geometria b	**Limite da Geometria b**	**Exterior da Geometria b**
Interior da Geometria a	T	*	*
Limite da Geometria a	*	*	*
Exterior da Geometria a	F	F	*

Exemplos

A Figura 19 na página 270 mostra exemplos de ST_Contains:

- Uma geometria multiponto contém geometrias de um ponto ou multiponto quando todos os pontos estão dentro da primeira geometria.
- Uma geometria de polígono contém uma geometria multiponto quando todos os pontos estão no limite do polígono ou no interior do polígono.
- Uma geometria de sequência de linhas contém geometrias de um ponto, multiponto ou de sequência de linhas quando todos os pontos estão dentro da primeira geometria.
- Uma geometria de polígono contém geometrias de um ponto ou de sequência de linhas ou de polígono quando a segunda geometria está no interior do polígono.

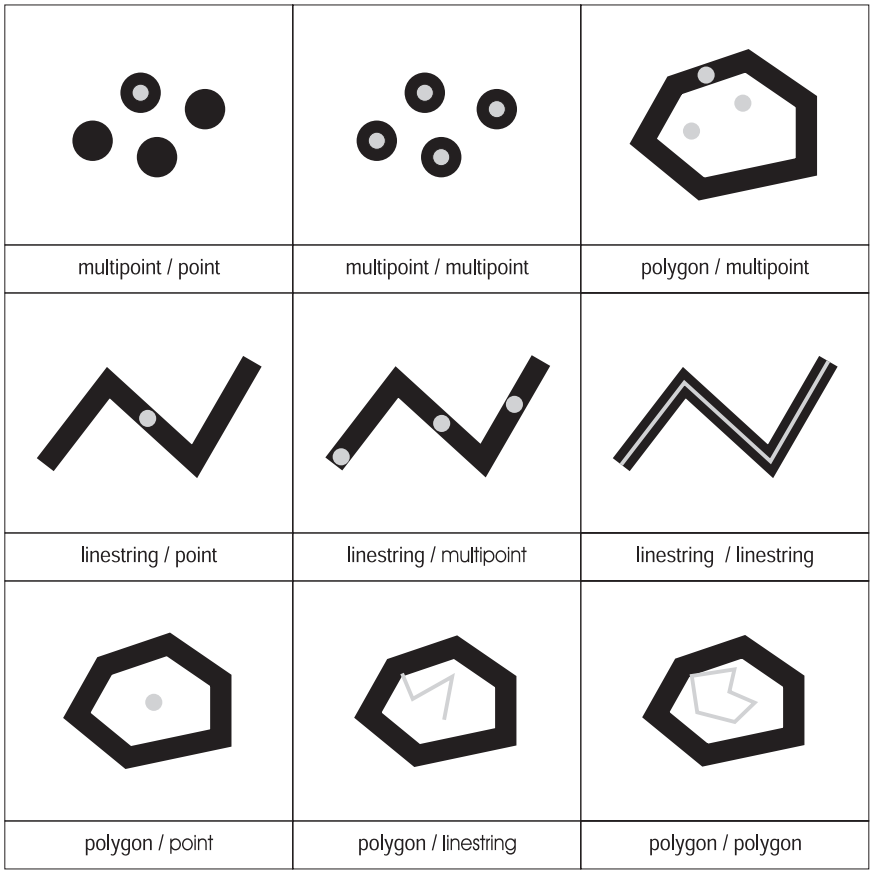


Figura 19. ST_Contains. As geometrias escuras representam a geometria "a" e as geometrias cinzas representam a geometria "b". Em todos os casos, a geometria a contém a geometria b completamente.

Exemplo 1

O código a seguir cria e ocupa estas tabelas.

```
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) )
```

Exemplo 2

O fragmento de código a seguir utiliza a função ST_Contains para determinar quais pontos são contidos por um polígono específico.

```
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
```

Exemplo 3

O fragmento de código a seguir utiliza a função ST_Contains para determinar quais linhas são contidas por um polígono específico.

```
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
```

Função ST_ConvexHull

A função ST_ConvexHull usa uma geometria como um parâmetro de entrada e retorna o envoltório convexo dela.

A geometria resultante é representada no sistema de referência espacial da geometria especificada.

Se possível, o tipo específico da geometria retornada será ST_Point, ST_LineString ou ST_Polygon. Por exemplo, o limite de um polígono sem aberturas é uma sequência de linhas única, representada como ST_LineString. O limite de um polígono com uma ou mais aberturas consiste em várias sequências de linhas, representadas como ST_MultiLineString.

Se a geometria especificada for nula ou vazia, então nulo é retornado.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria para calcular o envoltório convexo.

Tipo de retorno

db2gse.ST_Geometry

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

O código a seguir cria e ocupa a tabela 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))
```

A instrução SELECT a seguir calcula o envoltório convexo para todas as geometrias construídas anteriormente e exibe o 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))
```

Função ST_CoordDim

A função ST_CoordDim usa uma geometria como um parâmetro de entrada e retorna a dimensionalidade de suas coordenadas.

Se a geometria especificada não tiver coordenadas Z e M, a dimensão será 2. Se tiver coordenadas Z e nenhuma coordenada M, ou se tiver coordenadas M e nenhuma coordenada Z, a dimensão será 3. Se tiver coordenadas Z e M, a dimensão será 4. Se a geometria for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria a partir da qual a dimensão será recuperada.

Tipo de retorno

INTEGER

Exemplo

Este exemplo cria várias geometrias e depois determina a dimensão de suas 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
```

Função ST_Crosses

A função ST_Crosses usa duas geometrias como parâmetros de entrada e retorna 1 se a primeira geometria cruzar a segunda. Caso contrário, será retornado 0 (zero).

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria, ela será convertida para o outro sistema de referência espacial.

Se a primeira geometria for um polígono ou um multipolígono, ou se a segunda geometria for um ponto ou um multiponto ou se qualquer uma das geometrias for um valor nulo ou estiver vazia, será retornado nulo. Se a interseção das duas geometrias resultar em uma geometria que tenha uma dimensão menor do que a dimensão máxima das duas geometrias especificadas e se a geometria resultante não for igual à nenhuma das duas geometrias especificadas, será retornado 1. Caso contrário, o resultado será 0 (zero).

Sintaxe

Parâmetro

geometry1
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que deve ser testada para cruzar *geometry2*.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que deve ser testada para determinar se ela é cruzada por *geometry1*.

Tipo de Retorno

INTEGER

Exemplo

Este código determina se as geometrias construídas se cruzam.

```
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
```

Função ST_Difference

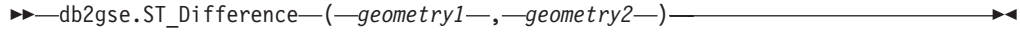
A função ST_Difference usa duas geometrias como parâmetros de entrada e retorna a parte da primeira geometria que não cruza com a segunda geometria.

As duas geometrias devem ter a mesma dimensão. Se qualquer uma das geometrias for nula, será retornado nulo. Se a primeira geometria for vazia, será retornada uma geometria vazia do tipo ST_Point. Se a segunda geometria for vazia, será retornada a primeira geometria inalterada.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria e usar o mesmo datum subjacente, ela será convertida no outro sistema de referência espacial.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometry1
Um valor do tipo ST_Geometry que representa a primeira geometria a ser utilizada para calcular a diferença para *geometry2*.

geometry2
Um valor do tipo ST_Geometry que representa a segunda geometria que é utilizada para calcular a diferença para *geometry1*.

Tipo de retorno

db2gse.ST_Geometry

A dimensão da geometria retornada é igual à das geometrias de entrada.

Exemplos

No exemplo a seguir, os resultados foram reformatados para possibilitar a leitura. O espaçamento nos resultados irá variar de acordo com a exibição.

O código a seguir cria e ocupa a tabela 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))
```

Exemplo 1

Este exemplo encontra a diferença entre dois polígonos separados.

```
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))
```

Exemplo 2

Este exemplo encontra a diferença entre dois polígonos que fazem interseção.

```
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))
```

Exemplo 3

Este exemplo encontra a diferença entre duas sequências de linhas de sobreposição.

```
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)
```

Função ST_Dimension

A função ST_Dimension usa uma geometria como um parâmetro de entrada e retorna sua dimensão.

Se a geometria especificada for vazia, -1 será retornado. Para pontos e multipontos, a dimensão é 0 (zero); para curvas e multicurvas, a dimensão é 1; e para polígonos e multipolígonos, a dimensão é 2. Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry que representa a geometria para a qual a dimensão é retornada.

Tipo de retorno

INTEGER

Exemplo

Este exemplo cria várias geometrias diferentes e encontra suas dimensões.

```
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
```

Função ST_Disjoint

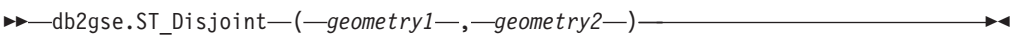
A função ST_Disjoint usa duas geometrias como parâmetros de entrada e retorna 1 se as geometrias especificadas não se cruzarem. Se as geometrias forem interseccionadas, será retornado 0 (zero).

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria, ela será convertida para o outro sistema de referência espacial.

Se qualquer uma das duas geometrias for nula ou vazia, será retornado um valor nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometry1
Um valor do tipo ST_Geometry que representa a geometria que é testada para ser separada de *geometry2*.

geometry2
Um valor do tipo ST_Geometry que representa a geometria que será testada para ser separada de *geometry1*.

Tipo de retorno

INTEGER

Exemplos

Exemplo 1

Este código cria várias geometrias na tabela 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))
```

Exemplo 2

Este exemplo determina se o primeiro polígono está separado de qualquer uma das geometrias.

```
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
```

Exemplo 3

Este exemplo determina se o terceiro polígono está separado de qualquer uma das geometrias.

```
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
```

Exemplo 4

Este exemplo determina se a segunda sequência de linhas está separada de qualquer uma das geometrias.

```
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
```

função ST_Distance

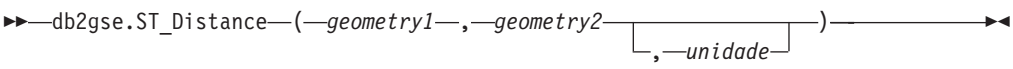
A função ST_Distance usa duas geometrias e, opcionalmente, uma unidade como parâmetros de entrada e retorna a menor distância entre qualquer ponto na primeira geometria para qualquer ponto na segunda geometria, medida nas unidades padrão ou especificadas.

Se qualquer uma das duas geometrias for nula ou vazia, será retornado nulo.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria e usar o mesmo datum subjacente, ela será convertida no outro sistema de referência espacial.

Também é possível chamar essa função como um método quando você oferece uma unidade de medida.

Sintaxe



Parâmetro

geometry1
Um valor do tipo ST_Geometry que representa a geometria que é utilizada para calcular a distância para *geometry2*.

geometry2
Um valor do tipo ST_Geometry que representa a geometria que é utilizada para calcular a distância para *geometry1*.

unit VARCHAR(128) valor que identifica a unidade na qual o resultado é medido. As unidades de medida suportadas estão listadas na exibição do catálogo DB2GSE.ST_UNITS_OF_MEASURE.

Se o parâmetro *unit* for omitido, as seguintes regras serão utilizadas para determinar a unidade de medida utilizada para o resultado:

- Se *geometry1* estiver em um sistema de coordenadas projetadas ou geocêntricas, a unidade linear associada a este sistema de coordenadas será a padrão.
- Se *geometry1* estiver em um sistema de coordenadas geográficas, a unidade angular associada a este sistema de coordenadas será o padrão.

Restrições para conversões de unidades: Será retornado um erro (SQLSTATE 38SU4) se ocorrer qualquer uma das condições a seguir:

- A geometria está em um sistema de coordenadas não especificado e o parâmetro *unit* é especificado.
- A geometria está em um sistema de coordenadas projetadas e uma unidade angular é especificada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

As instruções SQL a seguir criam e ocupam as tabelas SAMPLE_GEOMETRIES1 e 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) )
```

Exemplo 2

A seguinte instrução SELECT calcula a distância entre as diversas geometrias nas tabelas SAMPLE_GEOMTRIES1 e 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
```

Exemplo 3

A seguinte instrução SELECT ilustra como encontrar todas as geometrias que estão a uma distância de 100 umas das outras.

```
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
```

Exemplo 4

A seguinte instrução SELECT calcula a distância, em quilômetros, entre as diversas geometrias.

```
Tabelas SAMPLE_GEOMTRIES1 e 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
```

função ST_DistanceToPoint

A função ST_DistanceToPoint usa uma geometria de curva ou de multicurva e uma geometria de ponto como parâmetros de entrada e retorna a distância na geometria de curva para o ponto especificado.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria de curva
Um valor do tipo ST_Curve ou ST_MultiCurve ou um de seus subtipos que representa a geometria a ser processada.

geometria de ponto
Um valor do tipo ST_Point que é um ponto ao longo da curva especificada.

Tipo de retorno

DOUBLE

Exemplo

A seguinte instrução SQL cria a tabela SAMPLE_GEOMETRIES com duas colunas. A coluna do ID identifica exclusivamente cada linha. A coluna GEOMETRY ST_LineString armazena geometrias de amostra.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

A seguinte instrução SQL insere duas linhas na tabela SAMPLE_GEOMETRIES.

```
INSERT INTO sample_geometries(id, geometry)
VALUES
 (1,ST_LineString('LINESTRING ZM(0 0 0 0, 10 100 1000 10000)',1)),
 (2,ST_LineString('LINESTRING ZM(10 100 1000 10000, 0 0 0 0)',1))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente mostram como usar a função ST_DistanceToPoint para localizar a distância até o ponto no local (1.5, 15.0).

```
SELECT ID, DECIMAL(ST_DistanceToPoint(geometry,ST_Point(1.5,15.0,1)),10,5) AS DISTANCE
FROM   sample_geometries

ID          DISTANCE
----------- ------------
          1     15.07481
          2     85.42394

  2 record(s) selected.
```

Função ST_Endpoint

A função ST_Endpoint usa uma curva como um parâmetro de entrada e retorna o ponto que é o último ponto da curva. O ponto resultante é representado no sistema de referência espacial da curva especificada.

Se a curva especificada for nula ou vazia, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

curve Um valor do tipo ST_Curve que representa a geometria a partir da qual o último ponto é retornado.

Tipo de retorno

db2gse.ST_Point

Exemplo

A instrução SELECT encontra o ponto extremo de cada uma das geometrias na tabela 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)
```

Função ST_Envelope

A função ST_Envelope usa uma geometria como um parâmetro de entrada e retorna um envelope na geometria. O envelope é um retângulo que é representado como um polígono.

Se a geometria especificada for um ponto, uma sequência de linhas horizontal ou uma sequência de linhas vertical, será retornado um retângulo, que é ligeiramente maior do que a geometria especificada. Caso contrário, o retângulo limitador mínimo da geometria será retornado como envelope. Se a geometria especificada for nula ou vazia, então nulo é retornado. Para retornar o retângulo de limite mínimo exato para todas as geometrias, utilize a função ST_MBR.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry que representa a geometria para a qual o envelope será retornado.

Tipo de retorno

db2gse.ST_Polygon

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo cria várias geometrias e depois determina seus envelopes. Para o ponto não vazio e a sequência de linhas (que é horizontal), o envelope é um retângulo que é ligeiramente maior do que a geometria.

```
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))
```

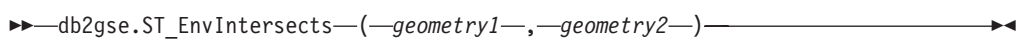
Função ST_EnvIntersects

A função ST_EnvIntersects usa duas geometrias como parâmetros de entrada e retorna 1 se os envelopes das duas geometrias se cruzarem. Caso contrário, será retornado 0 (zero).

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria, ela será convertida para o outro sistema de referência espacial.

Se uma das geometrias indicadas for nula ou vazia, o valor nulo será retornado.

Sintaxe



Parâmetro

geometry1
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria cujo envelope será testado quanto à interseção com o envelope de *geometry2*.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria cujo envelope será testado quanto à interseção com o envelope de *geometry1*.

Tipo de retorno

INTEGER

Exemplo

Este exemplo cria duas sequências de linha paralelas e verifica sua interseção. As próprias sequências de linha não fazem interseção, mas seus envelopes sim.

```
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
```

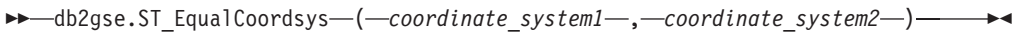
Função ST_EqualCoordsys

A função ST_EqualCoordsys usa duas definições do sistema de coordenadas como parâmetros de entrada e retorna o valor de número inteiro 1 (um) se as definições especificadas forem idênticas. Caso contrário, será retornado o valor inteiro 0 (zero).

As definições do sistema de coordenadas são comparadas independentemente das diferenças de espaços, parênteses, caracteres maiúsculos e minúsculos e a representação de números de ponto flutuante.

Se qualquer uma das definições especificadas do sistema de coordenadas for nula, será retornado nulo.

Sintaxe



Parâmetro

coordinate_system1
Um valor do tipo VARCHAR(2048) que define o primeiro sistema de coordenadas a ser comparado com *coordinate_system2*.

coordinate_system2
Um valor do tipo VARCHAR(2048) que define o segundo sistema de coordenadas a ser comparado com *coordinate_system1*.

Tipo de retorno

INTEGER

Exemplo

Este exemplo compara dois sistemas de coordenadas australianos para verificar se são iguais.

```
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
```

Função ST_Equals

A função ST_Equals usa duas geometrias como parâmetros de entrada e retorna 1 se as geometrias forem iguais. Caso contrário, será retornado 0 (zero). A ordem dos pontos utilizados para definir a geometria não é relevante para o teste de igualdade.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria, ela será convertida para o outro sistema de referência espacial.

Se qualquer uma das duas geometrias especificadas for nula, será retornado nulo.

Sintaxe

Parâmetro

geometry1
Um valor do tipo ST_Geometry que representa a geometria que será comparada com *geometry2*.

geometry2
Um valor do tipo ST_Geometry que representa a geometria que será comparada com *geometry1*.

Tipo de retorno

INTEGER

Exemplos

Exemplo 1

Este exemplo cria dois polígonos que têm suas coordenadas em uma ordem diferente. ST_Equal é utilizado para mostrar que estes polígonos são considerados iguais.

```
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
```

Exemplo 2

Neste exemplo, duas geometrias são criadas com as mesmas coordenadas X e Y, mas com coordenadas M diferentes (medidas). Quando as geometrias são comparadas com a função ST_Equal, é retornado um 0 (zero) para indicar que estas geometrias não são iguais.

```
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
```

Exemplo 3

Neste exemplo, são criadas duas geometrias com um conjunto de coordenadas diferentes, mas ambas representam a mesma geometria. ST_Equal compara as geometrias e indica que ambas são realmente iguais.

```
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:

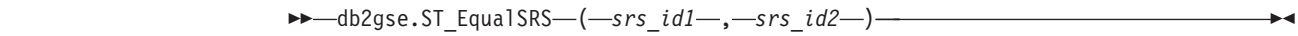
```
ID          ID          EQUALS
----------- ----------- -----------
          5           6           1
```

Função ST_EqualSRS

A função ST_EqualSRS usa dois identificadores de sistema de referência espacial como parâmetros de entrada e retorna 1 se os sistemas de referência espacial especificados forem idênticos. Caso contrário, será retornado 0 (zero). Os deslocamentos, fatores de escala e sistemas de coordenadas são comparados.

Se qualquer um dos identificadores especificados do sistema de referência espacial for nulo, será retornado nulo.

Sintaxe



Parâmetro

srs_id1
Um valor do tipo INTEGER que identifica o primeiro sistema de referência espacial a ser comparado com o sistema de referência espacial identificado por *srs_id2*.

srs_id2
Um valor do tipo INTEGER que identifica o segundo sistema de referência espacial a ser comparado com o sistema de referência espacial identificado por *srs_id1* .

Tipo de retorno

INTEGER

Exemplo

São criados dois sistemas de referência espacial similares com as seguintes chamadas para db2se.

```
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
```

Estes SRSs têm os mesmos valores de deslocamento e escala e se referem aos mesmos sistemas de coordenadas. A única diferença está no nome definido e no ID do SRS. Portanto, a comparação retorna 1, que indica que eles são iguais.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

VALUES ST_EqualSRS(12, 22)
```

Resultados:

```
1
-----------
          1
```

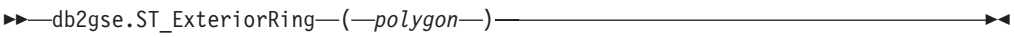
Função ST_ExteriorRing

A função ST_ExteriorRing usa um polígono como um parâmetro de entrada e retorna seu anel externo como uma curva. A curva resultante é representada como o sistema de referência espacial do polígono especificado.

Se o polígono especificado for nulo ou vazio, será retornado nulo. Se o polígono não tiver anéis internos, o anel externo retornado será idêntico ao limite do polígono.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

polígono
Um valor do tipo ST_Polygon que representa o polígono para o qual o anel externo será retornado.

Tipo de retorno

db2gse.ST_Curve

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo cria dois polígonos, um com dois anéis internos e outro sem anéis internos, em seguida, determina seus anéis externos.

```
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)
```

Função ST_FindMeasure ou ST_LocateAlong

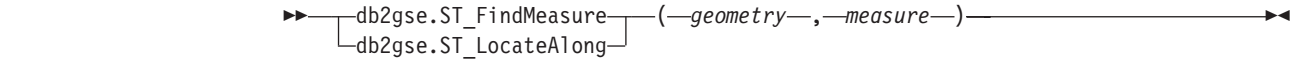
A função ST_FindMeasure ou ST_LocateAlong usa uma geometria e uma medida como parâmetros de entrada e retorna um multiponto ou multicurva dessa parte da geometria especificada que tem exatamente a medida especificada da geometria especificada que contém a medida especificada.

Para pontos e multipontos, todos os pontos com a medida especificada são retornados. Para curvas, multicurvas, superfícies e multisuperfícies, a interpolação é executada para calcular o resultado. O cálculo para superfícies e multisuperfícies é executado no limite da geometria.

Para pontos e multipontos, se a medida especificada não for encontrada, será retornada uma geometria vazia. Para as demais geometrias, se a medida especificada for menor do que a menor medida na geometria ou maior do que a maior medida da geometria, será retornada uma geometria vazia. Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria na qual serão procuradas partes cujas coordenadas M (medidas) contêm *measure*.

measure
Um valor do tipo DOUBLE que é a medida cujas partes da *geometria* devem ser incluídas no resultado.

Tipo de retorno

db2gse.ST_Geometry

Exemplos

Exemplo 1

A seguinte instrução CREATE TABLE cria a tabela SAMPLE_GEOMETRIES. SAMPLE_GEOMETRIES tem duas colunas: a coluna do ID, que identifica exclusivamente cada linha e a coluna GEOMETRY ST_Geometry, que armazena a geometria de exemplo.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id SMALLINT, geometry ST_GEOMETRY)
```

A seguinte instrução INSERT insere duas linhas. A primeira é uma sequência de linhas; a segunda é um multiponto.

```
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))
```

Exemplo 2

Na seguinte instrução SELECT e no conjunto de resultados correspondente, a função ST_FindMeasure é direcionada para localizar pontos cuja medida seja 7. A primeira linha retorna um ponto. No entanto, a segunda linha retorna um ponto vazio. Para recursos lineares (geometria com uma dimensão maior do que 0), ST_FindMeasure poderá interpolar o ponto; porém, para multipontos, a medida de destino deve ter uma correspondência exata.

```
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
```

Exemplo 3

Na seguinte instrução SELECT e no conjunto de resultados correspondente, a função ST_FindMeasure retorna um ponto e um multiponto. A medida de destino 6 corresponde às medidas nos dados de origem de ST_FindMeasure e no multiponto.

```
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)
```

Função ST_Generalize

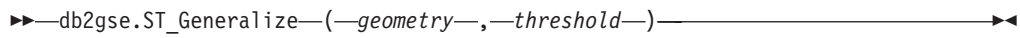
A função ST_Generalize usa uma geometria e um limite como parâmetros de entrada e representa a geometria especificada com um número reduzido de pontos, enquanto preserva as características gerais da geometria.

O algoritmo de simplificação de linha Douglas-Peucker é utilizado, através do qual a seqüência de pontos que definem a geometria é recursivamente subdivida até que uma sucessão dos pontos possa ser substituída por um segmento linear reto. Neste segmento de linha, nenhum dos pontos de definição é desviado do segmento de linha reto além do limite determinado. As coordenadas Z e M não são consideradas para a simplificação. A geometria resultante está no sistema de referência espacial da geometria especificada.

Se a geometria especificada for vazia, uma geometria vazia de tipo ST_Point será retornada. Se a geometria especificada ou o limite for nulo, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria à qual a simplificação de linha é aplicada.

threshold
Um valor do tipo DOUBLE que identifica o limite a ser utilizado para o algoritmo de simplificação de linha. O limite deve ser maior ou igual a 0 (zero). Quanto maior o limite, menor o número de pontos que serão utilizados para representar a geometria generalizada.

Tipo de retorno

db2gse.ST_Geometry

Exemplos

Nos exemplos a seguir, os resultados foram reformatados para possibilitar a leitura. O espaçamento nos resultados irá variar de acordo com a exibição.

Exemplo 1

Uma cadeia de linhas é criada com oito pontos que vão de (10, 10) a (80, 80). O caminho é quase uma linha reta, mas alguns dos pontos estão um pouco fora da linha. A função ST_Generalize pode ser utilizada para reduzir o número de pontos na linha.

```
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))
```

Exemplo 2

Quando um fator de generalização 3 é utilizado, a sequência de linhas é reduzida para quatro coordenadas e ainda fica muito próxima da representação original da sequência de linhas.

```
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)
```

Exemplo 3

Quando é utilizado um fator de generalização 6, a sequência de linhas é reduzida para somente duas coordenadas. Isto gera uma sequência de linhas mais simples do que o exemplo anterior, porém, se desvia mais da representação 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)
```

Função ST_GeomCollection

Use a função ST_GeomCollection para construir uma coleção de geometrias.

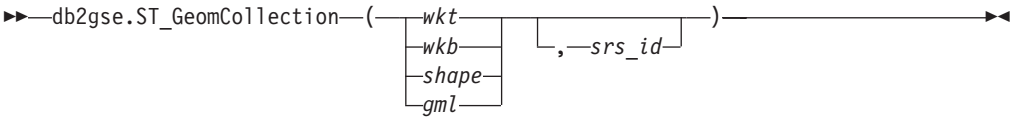
ST_GeomCollection constrói uma coleção de geometrias a partir de uma das seguintes entradas:

- Uma representação de texto reconhecida
- Uma representação binária reconhecida
- Uma representação de formatos ESRI
- Uma representação na linguagem de marcação geográfica (GML)

Um identificador do sistema de referência espacial opcional pode ser especificado para identificar o sistema de referência espacial no qual a coleção de geometrias resultante está localizado.

Se a representação de texto reconhecida, a representação binária reconhecida, a representação de formatos ESRI ou a representação GML for nula, será retornado nulo.

Sintaxe



Parâmetro

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida da coleção de geometrias resultantes.

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida da coleção de geometrias resultante.

shape Um valor do tipo BLOB(2G) que representa a representação de formatos ESRI da coleção de geometrias resultantes.

gml Um valor do tipo CLOB(2G) que representa a coleção de geometrias resultantes utiizando a linguagem de marcação geográfica (GML).

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial da coleta de geometria resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado implicitamente.

Se *srs_id* não identifica um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então um erro é retornado (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_GeomCollection

Notas

Se o parâmetro *srs_id* for omitido, poderá ser necessário converter *wkt* e *gml* explicitamente no tipo de dados CLOB. Caso contrário, o DB2 pode ser resolvido para a função utilizada para converter valores do tipo de referência REF(ST_GeomCollection) no tipo ST_GeomCollection. O exemplo a seguir assegura que o DB2 será resolvido para a função correta:

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

O código a seguir ilustra como a função ST_GeomCollection pode ser utilizada para criar e inserir um multiponto, multilinha e multipolígono a partir de representação WKT (well-known text) e um multiponto a partir da linguagem de marcação geográfica (GML) em uma coluna 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)
```

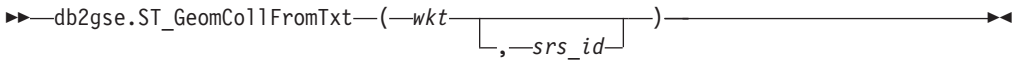
Função ST_GeomCollFromTxt

A função ST_GeomCollFromTxt usa uma representação de texto reconhecida de uma coleção de geometrias e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna a coleção de geometrias correspondente.

Se a representação de texto reconhecida for nula, então nulo é retornado.

A função recomendada para alcançar o mesmo resultado é ST_GeomCollection. Ela é recomendada por sua flexibilidade: ST_GeomCollection utiliza formatos adicionais de entrada, além da representação binária reconhecida.

Sintaxe



Parâmetro

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida da coleção de geometrias resultantes.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial da coleta de geometria resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado implicitamente.

Se *srs_id* não identifica um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então um erro é retornado (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_GeomCollection

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

O código a seguir ilustra como a função ST_GeomCollFromTxt pode ser utilizada para criar e inserir um multiponto, multilinha e um multipolígono a partir de uma representação de texto reconhecida (WKT) em uma coluna 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)))
```

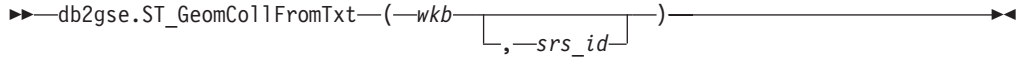
Função ST_GeomCollFromWKB

A função ST_GeomCollFromWKB usa uma representação binária reconhecida de uma coleção de geometrias e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna a coleção de geometrias correspondente.

Se representação binária reconhecida especificada for nula, então nulo é retornado.

A versão preferida para esta funcionalidade é ST_GeomCollection.

Sintaxe



Parâmetro

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida da coleção de geometrias resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial da coleta de geometria resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado implicitamente.

Se *srs_id* não identifica um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então um erro é retornado (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_GeomCollection

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

O código a seguir ilustra como a função ST_GeomCollFromWKB pode ser utilizada para criar e consultar as coordenadas de uma coleção de geometrias em uma representação binária reconhecida. As linhas são inseridas na tabela SAMPLE_GEOMCOLLECTION com IDs 4021 e 4022 e as coleções de geometrias no sistema de referência 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))
```

Função ST_Geometry

A função ST_Geometry constrói uma geometria a partir de uma representação especificada.

ST_Geometry constrói uma geometria a partir de uma das seguintes entradas:

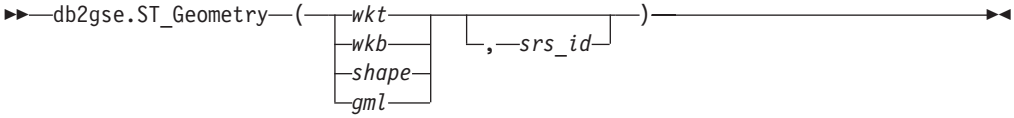
- Uma representação de texto reconhecida
- Uma representação binária reconhecida
- Uma representação de formatos ESRI
- Uma representação na linguagem de marcação geográfica (GML)

Um identificador do sistema de referência espacial opcional pode ser especificado para identificar o sistema de referência espacial no qual a geometria resultante está localizada.

O tipo dinâmico da geometria resultante é um dos subtipos instanciáveis de ST_Geometry.

Se a representação de texto reconhecida, a representação binária reconhecida, a representação de formatos ESRI ou a representação GML for nula, será retornado nulo.

Sintaxe



Parâmetro

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida da geometria resultante.

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida da geometria resultante.

shape Um valor do tipo BLOB(2G) que representa a representação de formatos ESRI da geometria resultante.

gml Um valor do tipo CLOB(2G) que representa a geometria resultante utilizando a linguagem de marcação geográfica (GML).

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial da geometria resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado implicitamente.

Se *srs_id* não identifica um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então um erro é retornado (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_Geometry

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

O código a seguir ilustra como a função ST_Geometry pode ser utilizada para criar e inserir um ponto a partir de uma representação de ponto de texto reconhecida (WKT) ou linha de uma representação de linha de Linguagem de Marcação Geográfica (GML).

A função ST_Geometry é a mais flexível das funções do construtor de tipos espaciais porque ela pode criar qualquer tipo espacial a partir de várias representações geométricas. ST_LineFromText pode criar apenas uma linha a partir da representação de linha WKT. ST_WKTToSql pode construir qualquer tipo, mas somente a partir da representação 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)
```

Função ST_GeometryN

A função ST_GeometryN usa uma coleção de geometrias e um índice como parâmetros de entrada e retorna a geometria na coleção identificada pelo índice. A geometria resultante é representada no sistema de referência espacial da coleção de geometria especificada.

Se a coleção de geometrias especificada for nula ou vazia ou se o índice for menor do que 1 ou maior do que o número de geometrias na coleção, será retornado nulo e ocorrerá uma condição de aviso (01HS0).

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

coleta Um valor do tipo ST_GeomCollection ou um de seus subtipos que representa a coleção de geometrias para localizar a *ú*ltima geometria.

índice Um valor do tipo INTEGER que identifica a *ú*ltima geometria que deve ser retornada da *coleção*.

Se *index* for menor do que 1 ou maior do que o número de geometrias na coleção, será retornado nulo e também um aviso (SQLSTATE 01HS0).

Tipo de retorno

db2gse.ST_Geometry

Exemplo

O código a seguir ilustra como escolher a segunda geometria em uma coleção de geometrias.

```
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))
```

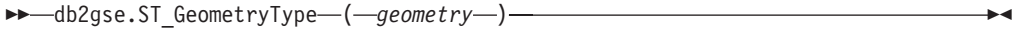
Função ST_GeometryType

A função ST_GeometryType usa uma geometria como parâmetro de entrada e retorna o nome completo do tipo do tipo dinâmico dessa geometria.

As funções TYPE_SCHEMA e TYPE_NAME do DB2 têm o mesmo efeito.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Geometry para o qual o tipo de geometria será retornado.

Tipo de retorno

VARCHAR(128)

Exemplos

O código a seguir ilustra como determinar o tipo de uma geometria.

```
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"
```

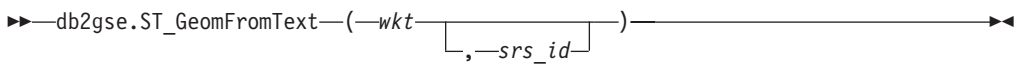
Função ST_GeomFromText

A função ST_GeomFromText usa uma representação de texto reconhecida de uma geometria e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna a geometria correspondente.

Se a representação de texto reconhecida for nula, então nulo é retornado.

A versão preferida para esta funcionalidade é ST_Geometry.

Sintaxe



Parâmetro

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida da geometria resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial da geometria resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado implicitamente.

Se *srs_id* não identifica um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então um erro é retornado (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_Geometry

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Neste exemplo, a função ST_GeomFromText é utilizada para criar e inserir um ponto a partir de uma representação de ponto de texto reconhecida (WKT).

O código a seguir insere linhas na tabela SAMPLE_POINTS com IDs e geometrias no sistema de referência espacial 1 utilizando a representação 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))
```

A instrução SELECT a seguir retornará o ID e GEOMETRIES a partir da tabela 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))
```

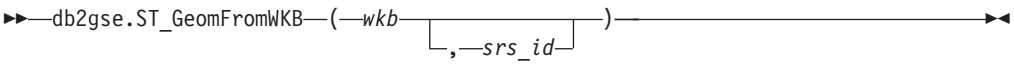
Função ST_GeomFromWKB

A função ST_GeomFromWKB usa uma representação binária reconhecida de uma geometria e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna a geometria correspondente.

Se representação binária reconhecida especificada for nula, então nulo é retornado.

A versão preferida para esta funcionalidade é ST_Geometry.

Sintaxe



Parâmetro

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida da geometria resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial da geometria resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado implicitamente.

Se o parâmetro srs_id especificado não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, será retornado um erro (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_Geometry

Exemplos

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

O código a seguir ilustra como a função ST_GeomFromWKB pode ser utilizada para criar e inserir uma linha a partir de uma representação de linha binária reconhecida (WKB).

O exemplo a seguir insere um registro na tabela SAMPLE_GEOMETRIES com um ID e uma geometria no sistema de referência espacial 1 em uma representação 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))
```

Funções Agregadas MBR

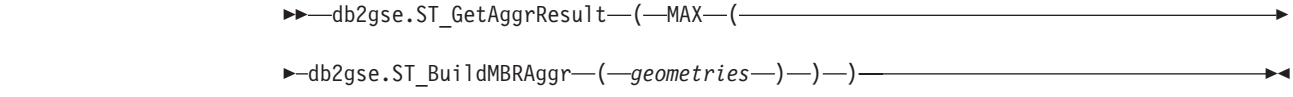
Use as funções ST_BuildMBRAggr e ST_GetAggrResult combinadas para agregar um conjunto de geometrias em uma coluna a uma única geometria. A combinação constrói um retângulo que representa o minimum bounding rectangle que inclui todas as geometrias na coluna. As coordenadas Z e M são descartadas quando o agregado é calculado.

A expressão a seguir é um exemplo que utiliza a função MAX com a função espacial db2gse.ST_BuildMBRAggr para calcular o MBR das geometrias na coluna columnName e a função espacial db2gse.ST_GetAggrResult para retornar a geometria resultante que foi calculada para o MBR:

```
db2gse.ST_Get_AggrResult(MAX(db2gse.ST_BuildMBRAggr(columnName)))
```

Se todas as geometrias a serem combinadas forem nulas, será retornado nulo. Se todas as geometrias forem nulas ou vazias, será retornada uma geometria vazia. Se o retângulo limite mínimo de todas as geometrias a serem combinadas resultar em um ponto, este ponto será retornado como valor ST_Point. Se o retângulo limite mínimo de todas as geometrias a serem combinadas resultar em uma sequência de linha horizontal ou vertical, essa sequência de linha será retornada como um valor ST_LineString. Caso contrário, o retângulo limite mínimo será retornado como um valor ST_Polygon.

Sintaxe



Parâmetro

MVS/ESA

Uma coluna selecionada do tipo ST_Geometry ou um dos seus subtipos e que representa todas as geometrias para as quais o retângulo limite mínimo deve ser calculado.

Tipo de retorno

db2gse.ST_Geometry

Restrições

Não é possível construir o agregado de união de uma coluna espacial em uma seleção completa em qualquer uma das seguintes situações:

- Em um ambiente de banco de dados particionado
- Se a cláusula GROUP BY for utilizada na seleção completa
- Se você utilizar uma função diferente da função agregada do DB2 MAX.

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo mostra como utilizar a função ST_BuildMBRAggr para obter o retângulo limite máximo de todas as geometrias dentro de uma coluna. Neste exemplo, diversos pontos são incluídos na coluna GEOMETRY na tabela SAMPLE_POINTS. Em seguida, o código SQL determina o retângulo limite máximo de todos os pontos 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))
```

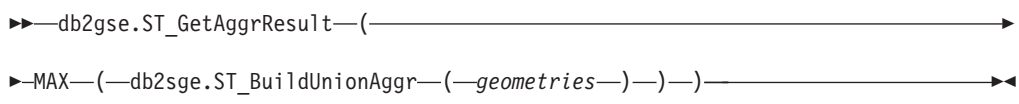
Funções Agregadas de União

Um agregado de união é a combinação das funções ST_BuildUnionAggr e ST_GetAggrResult. Use esta combinação para agregar uma coluna de geometrias em uma tabela a uma única geometria, construindo a união.

Se todas as geometrias a serem combinadas na união forem nulas, será retornado nulo. Se cada uma das geometrias a serem combinadas na união forem nulas ou vazias, será retornada uma geometria vazia do tipo ST_Point.

A função ST_BuildUnionAggr também pode ser chamada como um método.

Sintaxe



Parâmetros

MVS/ESA

Uma coluna em uma tabela do tipo ST_Geometry ou um dos seus subtipos e representa todas as geometrias que devem ser combinadas em uma união.

Tipo de retorno

db2gse.ST_Geometry

Restrições

Não é possível construir o agregado de união de uma coluna espacial em uma tabela em qualquer uma das seguintes situações:

- Em ambientes de banco de dados particionado
- Se uma cláusula GROUP BY for utilizada na seleção
- Se você utilizar uma função diferente da função agregada do DB2 MAX

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como um agregado de união pode ser utilizado para combinar um conjunto de pontos em multipontos. Diversos pontos são incluídos na tabela SAMPLE_POINTS. As funções ST_GetAggrResult e ST_BuildUnionAggr são utilizadas para construir a união dos pontos.

```
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
```

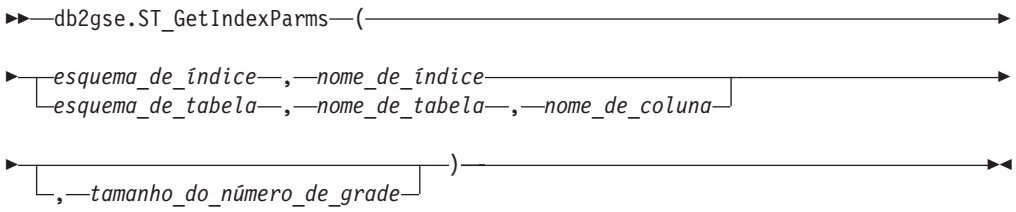
Resultados:

```
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)
```

Função ST_GetIndexParms

A função ST_GetIndexParms usa o identificador para um índice espacial ou para uma coluna espacial como um parâmetro de entrada e retorna os parâmetros usados para definir o índice ou o índice na coluna espacial. Se for especificado um número de parâmetro adicional, somente o tamanho de grade identificado pelo número será retornado.

Sintaxe



Parâmetro

index_schema
Um valor do tipo VARCHAR(128) que identifica o esquema no qual o índice espacial com o nome não qualificado *index_name* está. O nome do esquema faz distinção entre maiúsculas e minúsculas e deve ser listado na exibição do catálogo SYSCAT.SCHEMATA.

Se este parâmetro for nulo, o valor do registro especial CURRENT SCHEMA será utilizado como o nome do esquema para o índice espacial.

index_name
Um valor do tipo VARCHAR(128) que contém o nome não qualificado do índice espacial para o qual os parâmetros do índice são retornados. O nome do índice faz distinção entre maiúsculas e minúsculas e deve ser listado na exibição do catálogo SYSCAT.INDEXES para o esquema *index_schema*.

table_schema
Um valor do tipo VARCHAR(128) que identifica o esquema no qual a tabela com o nome não qualificado *table_name* está. O nome do esquema faz distinção entre maiúsculas e minúsculas e deve ser listado na exibição do catálogo SYSCAT.SCHEMATA.

Se este parâmetro for nulo, o valor do registro especial CURRENT SCHEMA será utilizado como o nome do esquema para o índice espacial.

table_name
Um valor do tipo VARCHAR(128) que contém o nome não qualificado da tabela com a coluna espacial *column_name*. O nome da tabela faz distinção entre maiúsculas e minúsculas e deve ser listado na exibição do catálogo SYSCAT.TABLES para o esquema *table_schema*.

column_name
Um valor do tipo VARCHAR(128) que identifica a coluna na tabela *table_schema.table_name* para a qual os parâmetros de índice do índice espacial dessa coluna são retornados. O nome da coluna faz distinção entre maiúsculas e minúsculas e deve ser listado na exibição do catálogo SYSCAT.COLUMNS para a tabela *table_schema.table_name*.

Se não houver nenhum índice espacial definido na coluna, ocorrerá um erro (SQLSTATE 38SQ0).

grid_size_number
Um valor DOUBLE que identifica o parâmetro cujo valor ou valores devem ser retornados.

Se este valor for menor do que 1 ou maior do que 3, ocorrerá um erro (SQLSTATE 38SQ1).

Tipo de retorno

DOUBLE (se *grid_size_number* for especificado)

Se *grid_size_number* não for especificado, será retornada uma tabela com as duas colunas ORDINAL e VALUE. A coluna ORDINAL será do tipo INTEGER e a coluna VALUE será do tipo DOUBLE.

Se os parâmetros forem retornados para um índice de grade, a coluna ORDINAL conterá os valores 1, 2 e 3 para os tamanhos da primeira, segunda e terceira grades, respectivamente. A coluna VALUE contém os tamanhos de grades.

A coluna VALUE contém os respectivos valores para cada um dos parâmetros.

Exemplos

Exemplo 1

Este código cria uma tabela com uma coluna espacial e um 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)
```

A função ST_GetIndexParms pode ser utilizada para recuperar os valores para os parâmetros que foram utilizados quando o índice espacial foi criado.

Exemplo 2

Este exemplo mostra como recuperar os três tamanhos de grades para um índice de grade espacial separadamente, especificando de forma explícita qual parâmetro, identificado por seu número, deve ser retornado.

```
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCALIZAÇÃO', 1)
```

Resultados:

```
1
------------------------
  +1.00000000000000E+000
```

```
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCALIZAÇÃO', 2)
```

Resultados:

```
1
------------------------
  +1.00000000000000E+001
```

```
VALUES ST_GetIndexParms('SCH', 'IDX', 3)
```

Resultados:

```
1
------------------------
  +1.00000000000000E+003
```

Exemplo 3

Este exemplo mostra como recuperar todos os parâmetros de um índice de grade espacial. A função ST_GetIndexParms retorna uma tabela que indica o número do parâmetro e o tamanho da grade correspondente.

```
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'OFFICES', 'LOCALIZAÇÃO') ) 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
```

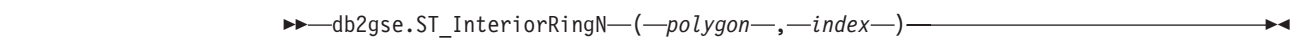
Função ST_InteriorRingN

A função ST_InteriorRingN usa um polígono e um índice como parâmetros de entrada e retorna o anel interno identificado pelo índice especificado como uma sequência de linhas. Os anéis internos são organizados de acordo com as regras definidas pelas rotinas de verificação de geometria interna.

Se o polígono fornecido for nulo ou vazio, ou se não tiver nenhum anel interno, será retornado nulo. Se o índice for menor do que 1 ou maior do que o número de anéis internos no polígono, será retornado nulo e ocorrerá uma condição de aviso (1HS1).

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

polígono
Um valor do tipo ST_Polygon que representa a geometria a partir da qual o anel interno identificado por *index* é retornado.

índice Um valor do tipo INTEGER que identifica o *ú*ltimo anel interno retornado. Se não houver nenhum anel interno identificado por *index*, ocorrerá uma condição de aviso (01HS1).

Tipo de retorno

db2gse.ST_Curve

Exemplo

Neste exemplo, é criado um polígono com dois anéis internos. A chamada ST_InteriorRingN é então utilizada para recuperar o segundo anel interno.

```
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)
```

Função ST_Intersection

A função ST_Intersection usa duas geometrias como parâmetros de entrada e retorna a geometria que é a intersecção das duas geometrias especificadas. A interseção é a parte comum da primeira geometria e da segunda geometria. A geometria resultante é representada no sistema de referência espacial da primeira geometria.

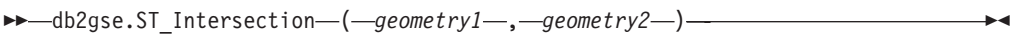
Se possível, o tipo específico da geometria retornada será ST_Point, ST_LineString ou ST_Polygon. Por exemplo, a interseção de um ponto e de um polígono é um ponto vazio ou único, representado como ST_MultiPoint.

Se qualquer uma das duas geometrias for nula, será retornado nulo.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria e usar o mesmo datum subjacente, ela será convertida no outro sistema de referência espacial.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometry1
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a primeira geometria para calcular a interseção com *geometry2*.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a segunda geometria para calcular a interseção com *geometry1*.

Tipo de retorno

db2gse.ST_Geometry

A dimensão da geometria retornada é a da entrada com a dimensão inferior.

Exemplo

Nos exemplos a seguir, os resultados foram reformatados para possibilitar a leitura. O espaçamento nos resultados irá variar de acordo com a exibição.

Este exemplo cria várias geometrias diferentes e depois determina a interseção (se houver) com a primeira.

```
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
```

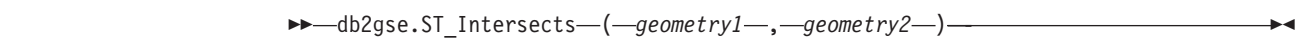
Resultados:

```
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 registros selecionados.
```

Função ST_Intersects

Use a função ST_Intersects para determinar se duas geometrias se cruzam.

Sintaxe



Parâmetro

geometry1
Um valor de tipo ST_Geometry ou um de seus subtipos que representa a geometria para testar a interseção com *geometry2*.

geometry2
Um valor de tipo ST_Geometry ou um de seus subtipos que representa a geometria para testar a interseção com *geometry1*.

Tipo de retorno

INTEGER

Uso

ST_Intersects utiliza duas geometrias como parâmetros de entrada e retorna 1 se as geometrias especificadas forem interseccionadas. Se as geometrias não se cruzarem, será retornado 0 (zero).

Se qualquer uma das duas geometrias for nula ou vazia, será retornado nulo.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria e usar o mesmo datum subjacente, ela será convertida no outro sistema de referência espacial.

A função ST_Intersects retorna o resultado oposto exato da função ST_Disjoint.

A função ST_Intersects retornará 1 (um) se as condições de qualquer uma das matrizes de padrão a seguir retornar TRUE.

Tabela 28. Matriz para ST_Intersects (1). A função ST_Intersects retornará 1 (um) se os interiores de ambas as geometrias se cruzarem.

	Interior da Geometria b	**Limite da Geometria b**	**Exterior da Geometria b**
Limite da Geometria a	*	*	*
Interior da Geometria a	T	*	*
Exterior da Geometria a	*	*	*

Tabela 29. Matriz para ST_Intersects (2). A função ST_Intersects retornará 1 (um) se o limite da primeira geometria cruzar o limite da segunda geometria.

	Interior da Geometria b	**Limite da Geometria b**	**Exterior da Geometria b**
Limite da Geometria a	*	*	*
Interior da Geometria a	*	T	*
Exterior da Geometria a	*	*	*

Tabela 30. Matriz para ST_Intersects (3). A função ST_Intersects retornará 1 (um) se o limite da primeira geometria cruzar o limite da segunda geometria.

	Interior da Geometria b	**Limite da Geometria b**	**Exterior da Geometria b**
Limite da Geometria a	T	*	*
Interior da Geometria a	*	*	*
Exterior da Geometria a	*	*	*

Tabela 31. Matriz para ST_Intersects (4). A função ST_Intersects retornará 1 (um) se os limites de uma das geometrias se cruzarem.

	Interior da Geometria b	**Limite da Geometria b**	**Exterior da Geometria b**
Limite da Geometria a	*	T	*
Interior da Geometria a	*	*	*
Exterior da Geometria a	*	*	*

Uso

Exemplo

As seguintes instruções criam e ocupam as tabelas SAMPLE_GEOMETRIES1 e 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) )
```

A seguinte instrução SELECT determina se as diversas geometrias nas tabelas SAMPLE_GEOMTRIES1 e SAMPLE_GEOMTRIES2 fazem interseção.

```
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 'Geometries do not intersect'
            WHEN 1 THEN 'Geometries intersect'
         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      Geometries intersect
     1 ST_Point         102 ST_Point      Geometries do not intersect
     1 ST_Point         103 ST_Point      Geometries do not intersect
     1 ST_Point         110 ST_LineString Geometries do not intersect
     1 ST_Point         120 ST_Polygon    Geometries do not intersect
     1 ST_Point         121 ST_Polygon    Geometries do not intersect
    10 ST_LineString    101 ST_Point      Geometries do not intersect
    10 ST_LineString    102 ST_Point      Geometries do not intersect
    10 ST_LineString    103 ST_Point      Geometries intersect
    10 ST_LineString    110 ST_LineString Geometries intersect
    10 ST_LineString    120 ST_Polygon    Geometries do not intersect
    10 ST_LineString    121 ST_Polygon    Geometries do not intersect
    20 ST_Polygon       101 ST_Point      Geometries intersect
    20 ST_Polygon       102 ST_Point      Geometries intersect
    20 ST_Polygon       103 ST_Point      Geometries do not intersect
    20 ST_Polygon       110 ST_LineString Geometries do not intersect
    20 ST_Polygon       120 ST_Polygon    Geometries intersect
    20 ST_Polygon       121 ST_Polygon    Geometries do not intersect
```

Função ST_Is3d

A função ST_Is3d usa uma geometria como um parâmetro de entrada e retorna 1 se a geometria especificada tiver coordenadas Z. Caso contrário, será retornado 0 (zero).

Se a geometria especificada for nula ou vazia, então nulo é retornado.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que será testada quanto à existência de coordenadas Z.

Tipo de retorno

INTEGER

Exemplo

Neste exemplo, são criadas várias geometrias com e sem coordenadas Z e coordenadas M (medidas). ST_Is3d é utilizada para determinar qual delas contém 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
```

Função ST_IsClosed

A função ST_IsClosed usa uma curva ou multicurva como um parâmetro de entrada e retorna 1 se a curva ou multicurva especificada estiver fechada. Caso contrário, será retornado 0 (zero).

Uma curva é fechada se o ponto de início e o ponto de término forem iguais. Se a curva tiver coordenadas Z, elas deverão ser iguais para os pontos de início e de

término. Caso contrário, os pontos não são considerados iguais e a curva não é fechada. Uma multicurva é fechada se cada uma de suas curvas for fechada.

Se a curva ou multicurva especificada for vazia, será retornado 0 (zero). Se for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

curve Um valor do tipo ST_Curve ou ST_MultiCurve ou um de seus subtipos que representa a curva ou multicurva que será testada.

Tipo de retorno

INTEGER

Exemplos

Exemplo 1

Este exemplo cria várias sequências de linhas. As duas últimas sequências de linhas têm as mesmas coordenadas X e Y, mas uma sequência de linha contém coordenadas Z variantes que impedem o fechamento da sequência de linhas e a outra sequência de linhas contém coordenas M variantes (medidas) que não afetam o fechamento da sequência de linhas.

```
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
```

Exemplo 2

Neste exemplo, são criadas duas sequências multilinha. ST_IsClosed é utilizado para determinar se as sequências multilinha são fechadas. A primeira não é fechada, mesmo que todas as curvas juntas formem um loop completo fechado. Isto ocorre porque cada curva por si não é fechada.

A segunda sequência multilinha é fechada porque cada curva por si é fechada.

```
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
```

Função ST_IsEmpty

A função ST_IsEmpty usa uma geometria como um parâmetro de entrada e retorna 1 se a geometria especificada estiver vazia. Caso contrário, será retornado 0 (zero).

Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que será testada.

Tipo de retorno

INTEGER

Exemplo

O código a seguir cria três geometrias e determina se são vazias.

```
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
```

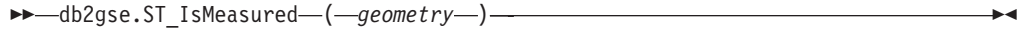
Função ST_IsMeasured

A função ST_IsMeasured usa uma geometria como um parâmetro de entrada. Se a geometria especificada tiver coordenadas M (medidas), ela retornará. Caso contrário, retornará 0 (zero).

Se a geometria especificada for nula ou vazia, então nulo é retornado.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria a ser testada quanto à existência de coordenadas M (medidas).

Tipo de retorno

INTEGER

Exemplo

Neste exemplo, são criadas várias geometrias com e sem coordenadas Z e coordenadas M (medidas). ST_IsMeasured é utilizada para determinar qual delas contém 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
```

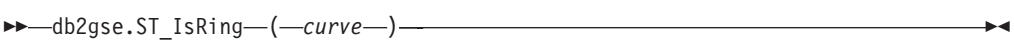
Função ST_IsRing

A função ST_IsRing usa uma curva como um parâmetro de entrada e retorna 1 se for um anel. Caso contrário, será retornado 0 (zero). Uma curva é um anel se for simples e fechada.

Se a curva especificada for vazia, então será retornado 0 (zero). Se for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

curve Um valor do tipo ST_Curve ou um de seus subtipos que representa a curva a ser testada.

Tipo de retorno

INTEGER

Exemplos

Neste exemplo, são criadas quatro sequências de linhas. ST_IsRing é utilizado para verificar se elas são anéis. A última não é considerada um anel mesmo que seja fechada, porque o caminho cruza sobre ela própria.

```
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
```

Função ST_IsSimple

A função ST_IsSimple usa uma geometria como um parâmetro de entrada e retorna 1 se a geometria especificada for simples. Caso contrário, será retornado 0 (zero).

Pontos, superfícies e multisuperfícies são sempre simples. Uma curva é simples se não passar duas vezes pelo mesmo ponto; um multiponto é simples se não tiver dois pontos iguais; e uma multicurva é simples se todas as suas curvas forem simples e as únicas interseções ocorrerem em pontos que estão no limite das curvas na multicurva.

Se a geometria especificada for vazia, será retornado o valor 1. Se for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que será testada.

Tipo de retorno

INTEGER

Exemplos

Neste exemplo, são criadas várias geometrias e é verificado se elas são simples. A geometria com um ID 4 não é considerada simples porque ela contém mais de um ponto que é igual. A geometria com um ID 6 não é considerada simples porque a sequência de linhas cruza com ela mesma.

```
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
```

Função ST_IsValid

A função ST_IsValid usa uma geometria como um parâmetro de entrada e retorna 1 se for válida. Caso contrário, será retornado 0 (zero).

Uma geometria somente será válida se todos os atributos no tipo estruturado forem consistentes com a representação interna de dados da geometria e se a representação interna não estiver danificada.

Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos.

Tipo de retorno

INTEGER

Exemplo

Este exemplo cria várias geometrias e utiliza ST_IsValid para verificar se elas são válidas. Todas as geometrias são válidas porque as rotinas do construtor, como ST_Geometry, não permitem a construção de geometrias inválidas.

```
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
```

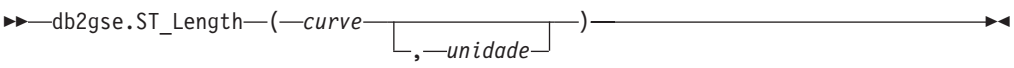
Função ST_Length

A função ST_Length usa uma curva ou multicurva e, opcionalmente, uma unidade como parâmetros de entrada e retorna o comprimento da curva ou multicurva especificada na unidade de medida padrão ou especificada.

Se a curva ou multicurva especificada for nula ou vazia, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

curve Um valor do tipo ST_Curve ou ST_MultiCurve que representa as curvas para as quais o comprimento é retornado.

unit Um valor VARCHAR(128) que identifica as unidades nas quais o comprimento da curva é medido. As unidades de medida suportadas estão listadas na exibição do catálogo DB2GSE.ST_UNITS_OF_MEASURE.

Se o parâmetro *unit* for omitido, as seguintes regras serão utilizadas para determinar a unidade na qual o comprimento é medido:

- Se *curve* estiver em um sistema de coordenadas projetadas ou geocêntricas, a unidade linear associada a este sistema de coordenadas será a padrão.
- Se *curve* estiver em um sistema de coordenadas geográficas, a unidade angular associada a este sistema de coordenadas será o padrão.

Restrições para conversões de unidades: Será retornado um erro (SQLSTATE 38SU4) se ocorrer qualquer uma das condições a seguir:

- A *curva* está em um sistema de coordenadas não especificado e o parâmetro *unit* é especificado.
- A *curva* está em um sistema de coordenadas projetadas e uma unidade angular é especificada.
- *curve* está em um sistema de coordenadas geográficas e uma unidade linear é especificada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

As instruções SQL a seguir criam uma tabela SAMPLE_GEOMETRIES e inserem uma linha e uma multilinha na tabela.

```
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))
```

Exemplo 2

A instrução SELECT a seguir calcula o comprimento da linha na tabela 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
```

Exemplo 3

A instrução SELECT a seguir calcula o comprimento da multilinha na tabela 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
```

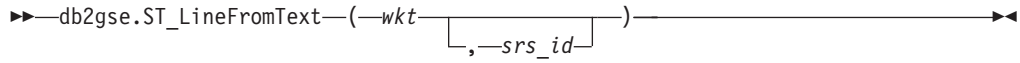
Função ST_LineFromText

A função ST_LineFromText usa uma representação de texto reconhecida de uma sequência de linhas e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna a sequência de linhas correspondente.

Se a representação de texto reconhecida for nula, então nulo é retornado.

A versão preferida para esta funcionalidade é ST_LineString.

Sintaxe



Parâmetro

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida da sequência de linhas resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial da sequência de linhas resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se *srs_id* não identifica um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então um erro é retornado (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_LineString

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

O código a seguir utiliza a função ST_LineFromText para criar e inserir uma linha a partir de uma representação de linha de texto reconhecida (WKT). As linhas são inseridas na tabela SAMPLE_LINES com um ID e um valor de linha no sistema de referência espacial 1 na representação 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
```

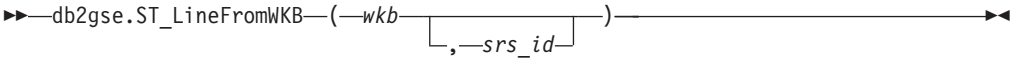
Função ST_LineFromWKB

A função ST_LineFromWKB usa uma representação binária reconhecida de uma sequência de linhas e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna a sequência de linhas correspondente.

Se representação binária reconhecida especificada for nula, então nulo é retornado.

A versão preferida para esta funcionalidade é ST_LineString.

Sintaxe



Parâmetro

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida da sequência de linhas resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial da sequência de linhas resultante.

Se o parâmetro ***srs_id*** for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se *srs_id* não identifica um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então um erro é retornado (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_LineString

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

O código a seguir utiliza a função ST_LineFromWKB para criar e inserir uma linha a partir de uma representação binária reconhecida. A linha é inserida na tabela SAMPLE_LINES com um ID e uma linha no sistema de referência espacial 1 na representação 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)
```

Função ST_LineString

A função ST_LineString constrói uma sequência de linhas a partir de uma entrada especificada.

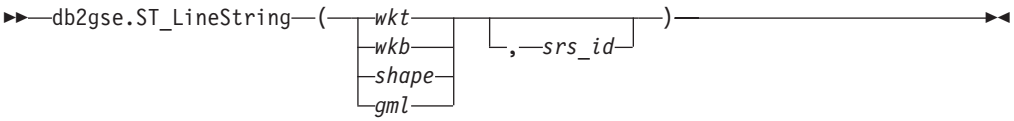
As entradas podem ser especificadas em um dos seguintes formatos:

- Uma representação de texto reconhecida
- Uma representação binária reconhecida
- Uma representação de formatos ESRI
- Uma representação na linguagem de marcação geográfica (GML)

Um identificador do sistema de referência espacial pode ser fornecido opcionalmente para identificar o sistema de referência espacial no qual a sequência de linhas resultante está localizada.

Se a representação de texto reconhecida, a representação binária reconhecida, a representação de formatos ESRI ou a representação GML for nula, será retornado nulo.

Sintaxe



Parâmetro

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida do polígono resultante.

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida do polígono resultante.

shape Um valor do tipo BLOB(2G) que representa a representação de formatos ESRI do polígono resultante.

gml Um valor do tipo CLOB(2G) que representa o polígono resultante utilizando a linguagem de marcação geográfica (GML).

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do polígono resultante.

Se o parâmetro *srs_id* for omitido, será utilizado o sistema de referência espacial com o identificador numérico 0 (zero).

Se *srs_id* não identificar um sistema de referência espacial listado na visualização de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, será retornado um erro (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_LineString

Exemplos

O código a seguir utiliza a função ST_LineString para criar e inserir uma linha a partir de uma representação de linha de WKT (Well-Know Text) ou a partir de uma representação WKB (Well-Know Binary).

O exemplo a seguir insere uma linha na tabela SAMPLE_LINES com um ID e linha no sistema de referência espacial 1 nas representações WKT e 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)
```

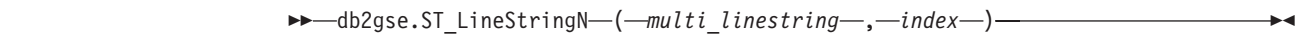
Função ST_LineStringN

A função ST_LineStringN usa uma sequência multilinha e um índice como parâmetros de entrada e retorna a sequência de linhas identificada pelo índice. A sequência de linhas resultante é representada no sistema de referência espacial da sequência multilinha especificada.

Se a sequência multilinha especificada for nula ou vazia, ou se o índice for menor do que 1 ou maior do que o número de sequências de linhas, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

multi_linestring
Um valor do tipo ST_MultiLineString que representa a sequência multilinha a partir da qual a sequência de linhas identificada por *index* é retornada.

índice Um valor do tipo INTEGER que identifica a *ú*ltima sequência de linhas, que deve ser retornada de *multi_linestring*.

Se *index* for menor do que 1 ou maior do que o número de sequências de linhas em *multi_linestring*, será retornado nulo e também uma condição de aviso (SQLSTATE 01HS0).

Tipo de retorno

db2gse.ST_LineString

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

A instrução SELECT ilustra como escolher a segunda geometria dentro de uma sequência multilinha na tabela 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)
```

Função ST_M

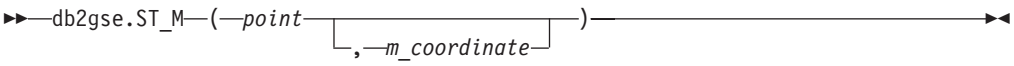
A função ST_M usa um ponto como um parâmetro de entrada e retorna sua coordenada de medida (M). Como opção, é possível indicar uma coordenada M como parâmetro de entrada além do ponto e a função retorna o próprio ponto com sua coordenada M configurada como o valor determinado.

Se a coordenada M especificada for nula, a coordenada M do ponto será removida.

Se o ponto especificado for nulo ou vazio, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

ponto Um valor do tipo ST_Point para o qual a coordenada M é retornada ou modificada.

m_coordinate
Um valor do tipo DOUBLE que representa a nova coordenada M para *point*.

Se *m_coordinate* for nulo, a coordenada M será removida de *point*.

Tipos de retornos

- DOUBLE, se *m_coordinate* não for especificado
- db2gse.ST_Point, se *m_coordinate* for especificado

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_M. Três pontos são criados e inseridos na tabela SAMPLE_POINTS. Todos eles estão no sistema de referência espacial que tem um ID 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))
```

Exemplo 2

Este exemplo encontra a coordenada M dos pontos na tabela 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
```

Exemplo 3

Este exemplo retorna um dos pontos com sua coordenada M definida como 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)
```

Função ST_MaxM

A função ST_MaxM usa uma geometria como um parâmetro de entrada e retorna sua coordenada M máxima.

Se a geometria especificada for nula ou vazia, ou se não tiver coordenadas M, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos para o qual a coordenada M máxima é retornada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_MaxM. Três polígonos são criados e inseridos na tabela 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) )
```

Exemplo 2

Este exemplo encontra a coordena M máxima de cada polígono em 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
```

Exemplo 3

Este exemplo encontra a coordenada M máxima existente para todos os polígonos na coluna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxM(geometry) ) AS INTEGER) OVERALL_MAX_M
  FROM sample_polys
```

Resultados:

```
OVERALL_MAX_M
-------------
           16
```

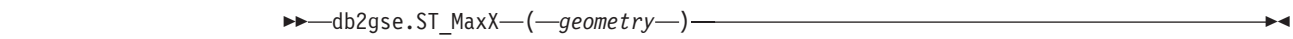
Função ST_MaxX

A função ST_MaxX usa uma geometria como um parâmetro de entrada e retorna sua coordenada X máxima.

Se a geometria especificada for nula ou vazia, então nulo é retornado.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos para o qual a coordenada X máxima é retornada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_MaxX. Três polígonos são criados e inseridos na tabela SAMPLE_POLYS. O terceiro exemplo ilustra como você pode utilizar todas as funções que retornam os valores de coordenas máxima e mínima para avaliar o intervalo espacial das geometrias que podem ser armazenadas em uma determinada coluna 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) )
```

Exemplo 2

Este exemplo encontra a coordena X máxima de cada polígono em 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
```

Exemplo 3

Este exemplo encontra a coordenada X máxima existente para todos os polígonos na coluna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxX(geometry) ) AS INTEGER) OVERALL_MAX_X
  FROM sample_polys
```

Resultados:

```
OVERALL_MAX_X
-------------
          120
```

Exemplo 4

Este exemplo encontra a extensão espacial (mínima total e máxima total) de todos os polígonos na tabela SAMPLE_POLYS. Este cálculo geralmente é utilizado para comparar a extensão espacial real das geometrias com a extensão do sistema de referência espacial associado aos dados para determinar se os dados podem ser expandidos.

```
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
```

Função ST_MaxY

A função ST_MaxY usa uma geometria como um parâmetro de entrada e retorna sua coordenada Y máxima.

Se a geometria especificada for nula ou vazia, então nulo é retornado.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos para o qual a coordenada Y máxima é retornada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_MaxY. Três polígonos são criados e inseridos na tabela 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) )
```

Exemplo 2

Este exemplo encontra a coordena Y máxima de cada polígono em 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
```

Exemplo 3

Este exemplo encontra a coordenada Y máxima existente para todos os polígonos na coluna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxY(geometry) ) AS INTEGER) OVERALL_MAX_Y
  FROM sample_polys
```

Resultados:

```
OVERALL_MAX_Y
--------------
           140
```

Função ST_MaxZ

A função ST_MaxZ usa uma geometria como um parâmetro de entrada e retorna sua coordenada Z máxima.

Se a geometria especificada for nula ou vazia, ou se não tiver coordenadas Z, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos para o qual a coordenada Z máxima é retornada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_MaxZ. Três polígonos são criados e inseridos na tabela 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) )
```

Exemplo 2

Este exemplo encontra a coordena Z máxima de cada polígono em 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
```

Exemplo 3

Este exemplo encontra a coordenada Z máxima existente para todos os polígonos na coluna GEOMETRY.

```
SELECT CAST ( MAX ( ST_MaxZ(geometry) ) AS INTEGER) OVERALL_MAX_Z
  FROM sample_polys
```

Resultados:

```
OVERALL_MAX_Z
-------------
           40
```

Função ST_MBR

A função ST_MBR usa uma geometria como um parâmetro de entrada e retorna seu minimum bounding rectangle.

Se a geometria especificada for um ponto, então o próprio ponto será retornado. Se a geometria for uma sequência de linhas horizontal ou uma sequência de linhas vertical, a própria sequência de linhas horizontal ou vertical será retornada. Se a geometria especificada for nula ou vazia, então nulo é retornado.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria à qual o retângulo de limite mínimo é retornado.

Tipo de retorno

db2gse.ST_Geometry

Exemplo

Este exemplo ilustra como a função ST_MBR pode ser utilizada para retornar o retângulo de limite mínimo de um polígono. Como a geometria especificada é um polígono, o retângulo de limite mínimo é retornado como um polígono.

Nos exemplos a seguir, as linhas de resultados foram formatadas novamente aqui para possibilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

```
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 ))
```

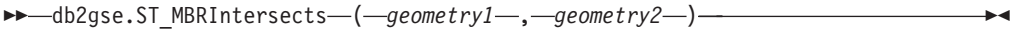
Função ST_MBRIntersects

A função ST_MBRIntersects usa duas geometrias como parâmetros de entrada e retorna 1 se os retângulos de limite mínimo das duas geometrias se cruzarem. Caso contrário, será retornado 0 (zero). O retângulo de limite mínimo de um ponto e uma sequência de linhas horizontal ou vertical representa a própria geometria.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria, ela será convertida para o outro sistema de referência espacial.

Se qualquer uma das geometrias especificadas for nula ou vazia, será retornado nulo.

Sintaxe



Parâmetros

geometry1
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria cujo retângulo de limite mínimo será testado para interseção com o retângulo de limite mínimo de *geometry2*.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria cujo retângulo de limite mínimo será testado para interseção com o retângulo de limite mínimo de *geometry1*.

Tipo de retorno

INTEGER

Exemplos

Exemplo 1

Este exemplo ilustra a utilização de ST_MBRIntersects para saber se dois polígonos que não fazem interseção estão próximos, verificando se seus retângulos de limite mínimo fazem interseção. O primeiro exemplo utiliza a expressão SQL CASE. O segundo exemplo utiliza uma única instrução SELECT para localizar os polígonos que fazem interseção com o retângulo de limite mínimo do polígono com ID = 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) )
```

Exemplo 2

A instrução SELECT a seguir utiliza uma expressão CASE para localizar os IDs dos polígonos que têm retângulos de limite mínimo que fazem interseção.

```
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
```

Exemplo 3

A instrução SELECT a seguir determina se os retângulos de limite mínimo para as geometrias fazem interseção com o polígono com ID = 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
```

Função ST_LocateBetween ou ST_MeasureBetween

A função ST_LocateBetween ou ST_MeasureBetween usa uma geometria e duas coordenadas M (medidas) como parâmetros de entrada e retorna essa parte da geometria especificada que representa o conjunto de caminhos ou pontos desconectados entre as duas coordenadas M.

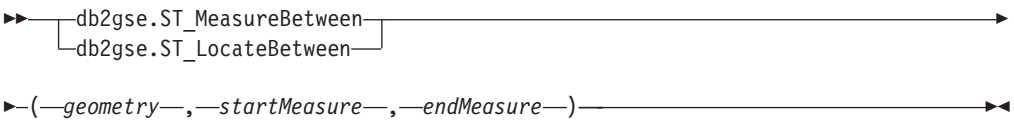
Para curvas, multicurvas, superfícies e multisuperfícies, a interpolação é executada para calcular o resultado. A geometria resultante é representada no sistema de referência espacial da geometria especificada.

Se a geometria especificada for uma superfície ou multisuperfície, ST_MeasureBetween ou ST_LocateBetween será aplicada aos anéis externo e interno da geometria. Se nenhuma das partes da geometria especificada estiver no intervalo definido pelas coordenadas M especificadas, será retornada uma geometria vazia. Se a geometria especificada for nula, então nulo é retornado.

Se a geometria resultante não estiver vazia, um tipo multiponto ou multilinhasequência será retornado.

As duas funções também podem ser chamadas como métodos.

Sintaxe



Parâmetros

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria na qual as partes com valores de medidas entre *startMeasure* e *endMeasure* devem ser encontradas.

startMeasure
Um valor do tipo DOUBLE que representa o limite inferior do intervalo de medidas. Se este valor for nulo, não será aplicado nenhum limite inferior.

endMeasure
Um valor do tipo DOUBLE que representa o limite superior do intervalo de medidas. Se este valor for nulo, não será aplicado nenhum limite superior.

Tipo de retorno

db2gse.ST_Geometry

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

A coordenada M (medida) de uma geometria é definida pelo usuário. Ela é muito versátil porque pode representar qualquer coisa que você deseja medir; por exemplo, a distância em uma rodovia, temperatura, pressão ou medidas de pH.

Este exemplo ilustra a utilização da coordenada M para registrar dados coletados de medidas de pH. Um pesquisador coleta o pH do solo em uma rodovia em locais específicos. Seguindo seus procedimentos operacionais padrão, ele anota os valores necessários em cada local do qual ele retira uma amostra do solo: as coordenadas X e Y desse local e o pH medido por ele.

```
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 o local em que a acidez do solo varia entre 4 e 6, o pesquisador utiliza esta instrução 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)
```

Função ST_LocateBetween ou ST_MeasureBetween

A função ST_LocateBetween ou ST_MeasureBetween usa uma geometria e duas coordenadas M (medidas) como parâmetros de entrada e retorna essa parte da geometria especificada que representa o conjunto de caminhos ou pontos desconectados entre as duas coordenadas M.

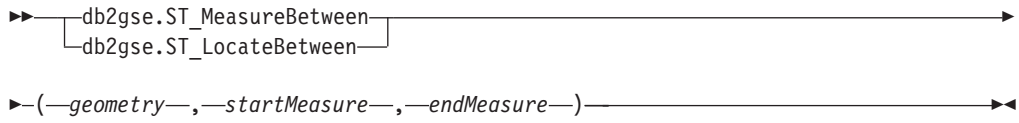
Para curvas, multicurvas, superfícies e multisuperfícies, a interpolação é executada para calcular o resultado. A geometria resultante é representada no sistema de referência espacial da geometria especificada.

Se a geometria especificada for uma superfície ou multisuperfície, ST_MeasureBetween ou ST_LocateBetween será aplicada aos anéis externo e interno da geometria. Se nenhuma das partes da geometria especificada estiver no intervalo definido pelas coordenadas M especificadas, será retornada uma geometria vazia. Se a geometria especificada for nula, então nulo é retornado.

Se a geometria resultante não estiver vazia, um tipo multiponto ou multilinhasequência será retornado.

As duas funções também podem ser chamadas como métodos.

Sintaxe



Parâmetros

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria na qual as partes com valores de medidas entre *startMeasure* e *endMeasure* devem ser encontradas.

startMeasure
Um valor do tipo DOUBLE que representa o limite inferior do intervalo de medidas. Se este valor for nulo, não será aplicado nenhum limite inferior.

endMeasure
Um valor do tipo DOUBLE que representa o limite superior do intervalo de medidas. Se este valor for nulo, não será aplicado nenhum limite superior.

Tipo de retorno

db2gse.ST_Geometry

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

A coordenada M (medida) de uma geometria é definida pelo usuário. Ela é muito versátil porque pode representar qualquer coisa que você deseja medir; por exemplo, a distância em uma rodovia, temperatura, pressão ou medidas de pH.

Este exemplo ilustra a utilização da coordenada M para registrar dados coletados de medidas de pH. Um pesquisador coleta o pH do solo em uma rodovia em locais específicos. Seguindo seus procedimentos operacionais padrão, ele anota os valores necessários em cada local do qual ele retira uma amostra do solo: as coordenadas X e Y desse local e o pH medido por ele.

```
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 o local em que a acidez do solo varia entre 4 e 6, o pesquisador utiliza esta instrução 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)
```

Função ST_MidPoint

A função ST_MidPoint usa uma curva como um parâmetro de entrada e retorna o ponto na curva que é equidistante dos dois pontos de extremidade da curva, medidos na curva. O ponto resultante é representado no sistema de referência espacial da curva especificada.

Se a curva especificada for vazia, então um ponto vazio será retornado. Se a curva especificada for nula, será retornado nulo.

Se a curva contiver coordenadas Z ou coordenadas M (medidas), o ponto médio será determinado somente pelos valores das coordenadas X e Y na curva. A coordenada Z e a medida no ponto retornado são interpoladas.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

curve Um valor do tipo ST_Curve ou um de seus subtipos que representa a curva para a qual o ponto no meio é retornado.

Tipo de retorno

db2gse.ST_Point

Exemplo

Este exemplo ilustra a utilização de ST_MidPoint para retornar o ponto médio 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)
```

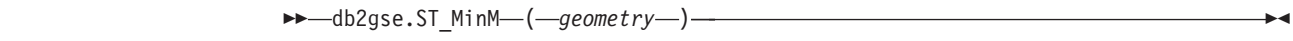
Função ST_MinM

A função ST_MinM usa uma geometria como um parâmetro de entrada e retorna sua coordenada M mínima.

Se a geometria especificada for nula ou vazia, ou se não tiver coordenadas M, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria

Um valor do tipo ST_Geometry ou um de seus subtipos para o qual a coordenada M mínima é retornada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_MinM. Três polígonos são criados e inseridos na tabela 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) )
```

Exemplo 2

Este exemplo encontra a coordena M mínima de cada polígono em 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
```

Exemplo 3

Este exemplo encontra a coordenada M mínima existente para todos os polígonos na coluna GEOMETRY.

```
SELECT CAST ( MIN ( ST_MinM(geometry) ) AS INTEGER) OVERALL_MIN_M
  FROM sample_polys
```

Resultados:

```
OVERALL_MIN_M
-------------
            3
```

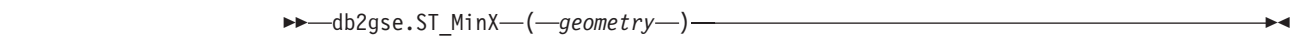
Função ST_MinX

A função ST_MinX usa uma geometria como um parâmetro de entrada e retorna sua coordenada X mínima.

Se a geometria especificada for nula ou vazia, então nulo é retornado.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos para o qual a coordenada X mínima é retornada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_MinX. Três polígonos são criados e inseridos na tabela 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) )
```

Exemplo 2

Este exemplo encontra a coordena X mínima de cada polígono em 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
```

Exemplo 3

Este exemplo encontra a coordenada X mínima existente para todos os polígonos na coluna GEOMETRY.

```
SELECT CAST ( MIN ( ST_MinX(geometry) ) AS INTEGER) OVERALL_MIN_X
  FROM sample_polys
```

Resultados:

```
OVERALL_MIN_X
-------------
            0
```

Função ST_MinY

A função ST_MinY usa uma geometria como um parâmetro de entrada e retorna sua coordenada Y mínima.

Se a geometria especificada for nula ou vazia, então nulo é retornado.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos para o qual a coordenada Y mínima é retornada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_MinY. Três polígonos são criados e inseridos na tabela 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) )
```

Exemplo 2

Este exemplo encontra a coordena Y mínima de cada polígono em 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
```

Exemplo 3

Este exemplo encontra a coordenada Y mínima existente para todos os polígonos na coluna GEOMETRY.

```
SELECT CAST ( MIN ( ST_MinY(geometry) ) AS INTEGER) OVERALL_MIN_Y
  FROM sample_polys
```

Resultados:

```
OVERALL_MIN_Y
--------------
            0
```

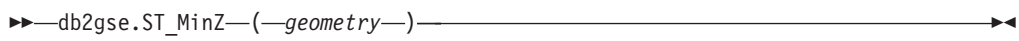
Função ST_MinZ

A função ST_MinZ usa uma geometria como um parâmetro de entrada e retorna sua coordenada Z mínima.

Se a geometria especificada for nula ou vazia, ou se não tiver coordenadas Z, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos para o qual a coordenada Z mínima é retornada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_MinZ. Três polígonos são criados e inseridos na tabela 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) )
```

Exemplo 2

Este exemplo encontra a coordena Z mínima de cada polígono em 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
```

Exemplo 3

Este exemplo encontra a coordenada Z mínima existente para todos os polígonos na coluna GEOMETRY.

```
SELECT CAST ( MIN ( ST_MinZ(geometry) ) AS INTEGER) OVERALL_MIN_Z
  FROM sample_polys
```

Resultados:

```
OVERALL_MIN_Z
-------------
             10
```

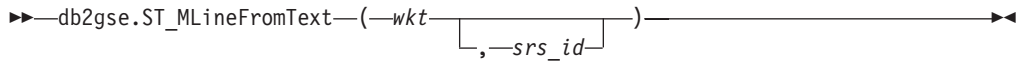
Função ST_MLineFromText

A função ST_MLineFromText usa uma representação de texto reconhecida de uma sequência multilinha e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna a sequência multilinha correspondente.

Se a representação de texto reconhecida for nula, então nulo é retornado.

A função recomendada para alcançar o mesmo resultado é a função ST_MultiLineString. Ela é recomendada por sua flexibilidade: ST_MultiLineString utiliza formatos adicionais de entrada, além da representação de texto reconhecida.

Sintaxe



Parâmetros

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida da sequência multilinha resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial para a sequência multilinha resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se o *srs_id* especificado não identificar um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então uma condição de exceção é criada (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_MultiLineString

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_MLineFromText pode ser utilizado para criar e inserir uma sequência multilinha a partir de sua representação de texto reconhecida. O registro inserido tem ID = 1110 e a geometria é uma sequência multilinha no sistema de referência espacial 1. A sequência multilinha está na representação de texto reconhecida de uma sequência multilinha. As coordenadas X e Y desta geometria são:

- Linha 1: (33, 2) (34, 3) (35, 6)
- Linha 2: (28, 4) (29, 5) (31, 8) (43, 12)
- Linha 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) )
```

A instrução SELECT a seguir retorna a sequência multilinha que foi registrada na tabela:

```
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 ))
```

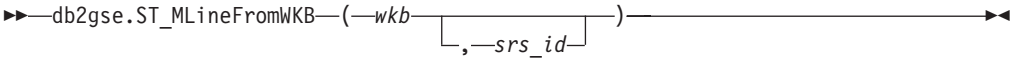
Função ST_MLineFromWKB

A função ST_MLineFromWKB usa uma representação binária reconhecida de uma sequência multilinha e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna a sequência multilinha correspondente.

Se representação binária reconhecida especificada for nula, então nulo é retornado.

A função recomendada para alcançar o mesmo resultado é a função ST_MultiLineString. Ela é recomendada por sua flexibilidade: ST_MultiLineString utiliza formatos adicionais de entrada, além da representação binária reconhecida.

Sintaxe



Parâmetros

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida da sequência multilinha resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial para a sequência multilinha resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se o *srs_id* especificado não identificar um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então uma condição de exceção é criada (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_MultiLineString

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_MLineFromWKB pode ser utilizado para criar uma sequência multilinha a partir de sua representação binária reconhecida. A geometria é uma sequência multilinha no sistema de referência espacial 1. Neste exemplo, a sequência multilinha é armazenada com ID = 10 na coluna GEOMETRY

da tabela SAMPLE_MLINES e, em seguida, a coluna WKB é atualizada com sua representação binária reconhecida (utilizando a função ST_AsBinary). Por último, a função ST_MLineFromWKB é utilizada para retornar a sequência multilinha da coluna WKB. As coordenadas X e Y desta geometria são:

- Linha 1: (61, 2) (64, 3) (65, 6)
- Linha 2: (58, 4) (59, 5) (61, 8)
- Linha 3: (69, 3) (67, 4) (66, 7) (68, 9)

A tabela SAMPLE_MLINES tem uma coluna GEOMETRY, em que a sequência multilinha é armazenada e uma coluna WKB, em que a representação binária reconhecida da sequência multilinha é armazenada.

```
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
```

Na instrução SELECT a seguir, a função ST_MLineFromWKB é utilizada para recuperar a sequência multilinha da coluna 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 ))
```

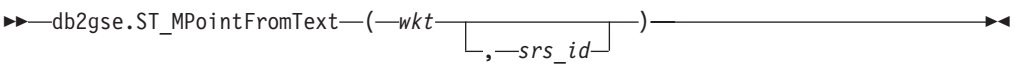
Função ST_MPointFromText

A função ST_MPointFromText usa uma representação de texto reconhecida de um multiponto e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna o multiponto correspondente.

Se a representação de texto reconhecida for nula, então nulo é retornado.

A função recomendada para alcançar o mesmo resultado é a função ST_MultiPoint. Ela é recomendada por sua flexibilidade: ST_MultiPoint utiliza formatos adicionais de entrada, além da representação de texto reconhecida.

Sintaxe



Parâmetros

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida do multiponto resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do multiponto resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se o *srs_id* especificado não identificar um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então uma condição de exceção é criada (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_MultiPoint

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_MPointFromText pode ser utilizado para criar e inserir um multiponto a partir de sua representação de texto reconhecida. O registro inserido tem ID = 1110 e a geometria é um multiponto no sistema de referência espacial 1. O multiponto está na representação de texto reconhecida de um multiponto. As coordenadas X e Y para esta geometria são: (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) )
```

A instrução SELECT a seguir retorna o multiponto que foi registrado na tabela:

```
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)
```

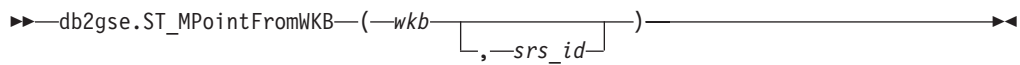
Função ST_MPointFromWKB

A função ST_MPointFromWKB usa uma representação binária reconhecida de um multiponto e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna o multiponto correspondente.

Se representação binária reconhecida especificada for nula, então nulo é retornado.

A função recomendada para alcançar o mesmo resultado é a função ST_MultiPoint. Ela é recomendada por sua flexibilidade: ST_MultiPoint utiliza formatos adicionais de entrada, além da representação binária reconhecida.

Sintaxe



Parâmetros

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida do multiponto resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do multiponto resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se o *srs_id* especificado não identificar um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então uma condição de exceção é criada (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_MultiPoint

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_MPointFromWKB pode ser utilizado para criar um multiponto a partir de sua representação binária reconhecida. A geometria é um multiponto no sistema de referência espacial 1. Neste exemplo, o multiponto é armazenado com ID = 10 na coluna GEOMETRY da tabela SAMPLE_MPOINTS e, em seguida, a coluna WKB é atualizada com sua representação binária reconhecida (utilizando a função ST_AsBinary). Por último, a função ST_MPointFromWKB é utilizada para retornar o multiponto da coluna WKB. As coordenadas X e Y para esta geometria são: (44, 14) (35, 16) (24, 13).

A tabela SAMPLE_MPOINTS tem uma coluna GEOMETRY, em que o multiponto é armazenado e uma coluna WKB, em que a representação binária reconhecida do multiponto é armazenada.

```
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
```

Na instrução SELECT a seguir, a função ST_MPointFromWKB é utilizada para recuperar o multiponto da coluna 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)
```

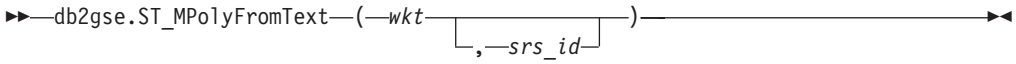
Função ST_MPolyFromText

A função ST_MPolyFromText usa uma representação de texto reconhecida de um multipolígono e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna o multipolígono correspondente.

Se a representação de texto reconhecida for nula, então nulo é retornado.

A função recomendada para alcançar o mesmo resultado é a função ST_MultiPolygon. Ela é recomendada por sua flexibilidade: ST_MultiPolygon utiliza formatos adicionais de entrada, além da representação de texto reconhecida.

Sintaxe



Parâmetros

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida do multipolígono resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do multipolígono resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se o *srs_id* especificado não identificar um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então uma condição de exceção é criada (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_MultiPolygon

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_MPolyFromText pode ser utilizado para criar e inserir um multipolígono a partir de sua representação de texto reconhecida. O registro inserido tem ID = 1110 e a geometria é um multipolígono no sistema de referência espacial 1. O multipolígono está na representação de texto reconhecida de um multipolígono. As coordenadas X e Y desta geometria são:

- 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) )
```

A instrução SELECT a seguir retorna o multipolígono que foi registrado na tabela:

```
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)))
```

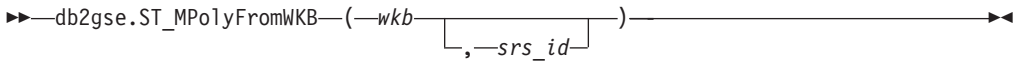
Função ST_MPolyFromWKB

A função ST_MPolyFromWKB usa uma representação binária reconhecida de um multipolígono e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna o multipolígono correspondente.

Se representação binária reconhecida especificada for nula, então nulo é retornado.

A função recomendada para alcançar o mesmo resultado é a função ST_MultiPolygon. Ela é recomendada por sua flexibilidade: ST_MultiPolygon utiliza formatos adicionais de entrada, além da representação binária reconhecida.

Sintaxe



Parâmetros

wkb Um valor do tipo BLOB(2 G) que contém a representação binária reconhecida do multipolígono resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do multipolígono resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se o *srs_id* especificado não identificar um sistema de referência espacial listado na exibição de catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, então uma condição de exceção é criada (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_MultiPolygon

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_MPolyFromWKB pode ser utilizado para criar um multipolígono a partir de sua representação binária reconhecida. A geometria é um multipolígono no sistema de referência espacial 1. Neste exemplo, o multipolígono é armazenado com ID = 10 na coluna GEOMETRY da tabela SAMPLE_MPOLYS e, em seguida, a coluna WKB é atualizada com sua representação binária reconhecida (utilizando a função ST_AsBinary). Por último, a função ST_MPolyFromWKB é utilizada para retornar o multipolígono da coluna WKB. As coordenadas X e Y desta geometria são:

- 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)

A tabela SAMPLE_MPOLYS tem uma coluna GEOMETRY, em que o multipolígono é armazenado e uma coluna WKB, em que a representação binária reconhecida do multipolígono é armazenada.

```
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
```

Na instrução SELECT a seguir, a função ST_MPolyFromWKB é utilizada para recuperar o multipolígono da coluna 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 )))
```

Função ST_MultiLineString

A função ST_MultiLineString constrói uma sequência multilinha a partir de uma entrada especificada.

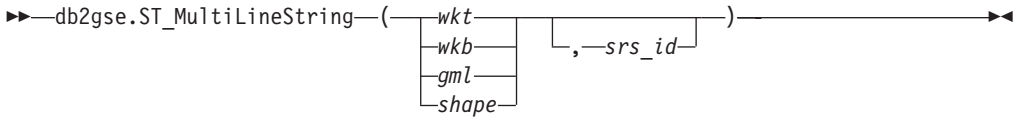
A entrada pode ser especificada em um dos seguintes formatos:

- Uma representação de texto reconhecida
- Uma representação binária reconhecida
- Uma representação de formatos
- Uma representação na linguagem de marcação geográfica (GML)

Um identificador do sistema de referência espacial opcional pode ser especificado para identificar o sistema de referência espacial no qual a sequência multilinha resultante está localizada.

Se a representação de texto reconhecida, a representação binária reconhecida, a representação de formatos ou a representação GML for nula, será retornado nulo.

Sintaxe



Parâmetros

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida da sequência multilinha resultante.

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida da sequência multilinha resultante.

gml Um valor do tipo CLOB(2G) que representa a sequência multilinha resultante utilizando a linguagem de marcação geográfica.

shape Um valor do tipo BLOB(2G) que especifica a representação de formatos da sequência multilinha resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial para a sequência multilinha resultante.

Se o parâmetro *srs_id* for omitido, será utilizado o sistema de referência espacial com o identificador numérico 0 (zero).

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_MultiLineString

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_MultiLineString pode ser utilizado para criar e inserir uma sequência multilinha a partir de sua representação de texto reconhecida. O registro inserido tem ID = 1110 e a geometria é uma sequência multilinha no sistema de referência espacial 1. A sequência multilinha está na representação de texto reconhecida de uma sequência multilinha. As coordenadas X e Y desta geometria são:

- Linha 1: (33, 2) (34, 3) (35, 6)
- Linha 2: (28, 4) (29, 5) (31, 8) (43, 12)
- Linha 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) )
```

A instrução SELECT a seguir retorna a sequência multilinha que foi registrada na tabela:

```
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 ))
```

Função ST_MultiPoint

A função ST_MultiPoint constrói um multiponto a partir de uma entrada especificada.

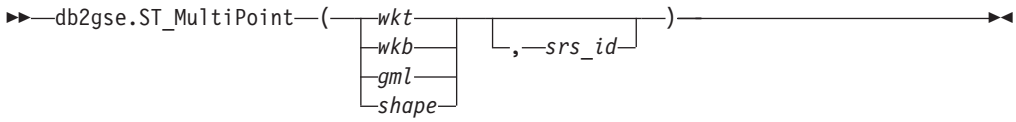
A entrada pode ser especificada em um dos seguintes formatos:

- Uma representação de texto reconhecida
- Uma representação binária reconhecida
- Uma representação de formatos
- Uma representação na linguagem de marcação geográfica (GML)

Um identificador do sistema de referência espacial opcional pode ser especificado para indicar o sistema de referência espacial no qual o multiponto resultante está localizado.

Se a representação de texto reconhecida, a representação binária reconhecida, a representação de formatos ou a representação GML for nula, será retornado nulo.

Sintaxe



Parâmetros

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida do multiponto resultante.

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida do multiponto resultante.

gml Um valor do tipo CLOB(2G) que representa o multiponto resultante utilizando a linguagem de marcação geográfica.

shape Um valor do tipo BLOB(2G) que especifica a representação de formatos do multiponto resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do multiponto resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_Point

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_MultiPoint pode ser utilizado para criar e inserir um multiponto a partir de sua representação de texto reconhecida. O registro inserido tem ID = 1110 e a geometria é um multiponto no sistema de referência espacial 1. O multiponto está na representação de texto reconhecida de um multiponto. As coordenadas X e Y para esta geometria são: (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))
```

A instrução SELECT a seguir retorna o multiponto que foi registrado na tabela:

```
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)
```

Função ST_MultiPolygon

ST_MultiPolygon constrói um multipolígono a partir de uma entrada especificada.

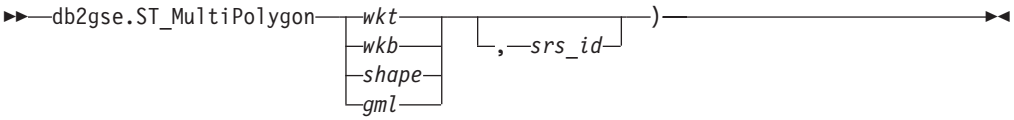
A entrada pode ser especificada em um dos seguintes formatos:

- Uma representação de texto reconhecida
- Uma representação binária reconhecida
- Uma representação de formatos
- Uma representação na linguagem de marcação geográfica (GML)

Um identificador do sistema de referência espacial opcional pode ser especificado para identificar o sistema de referência espacial no qual o multipolígono resultante está localizado.

Se a representação de texto reconhecida, a representação binária reconhecida, a representação de formatos ou a representação GML for nula, será retornado nulo.

Sintaxe



Parâmetros

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida do multipolígono resultante.

wkb Um valor do tipo BLOB(2 G) que contém a representação binária reconhecida do multipolígono resultante.

gml Um valor do tipo CLOB(2G) que representa o multipolígono resultante utilizando a linguagem de marcação geográfica.

shape Um valor do tipo BLOB(2G) que representa a representação de formatos do multipolígono resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do multipolígono resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_MultiPolygon

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_MultiPolygon pode ser utilizado para criar e inserir um multipolígono a partir de sua representação de texto reconhecida. O registro inserido tem ID = 1110 e a geometria é um multipolígono no sistema de referência espacial 1. O multipolígono está na representação de texto reconhecida de um multipolígono. As coordenadas X e Y desta geometria são:

- 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) )
```

A instrução SELECT a seguir retorna o multipolígono que foi registrado na tabela:

```
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)))
```

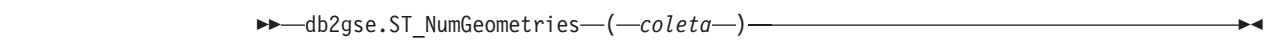
Função ST_NumGeometries

A função ST_NumGeometries usa uma coleção de geometrias como um parâmetro de entrada e retorna o número de geometrias na coleção.

Se a coleção de geometrias especificada for nula ou vazia, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

coleta Um valor do tipo ST_GeomCollection ou um de seus subtipos que representa a coleção de geometrias para a qual o número de geometrias é retornado.

Tipo de Retorno

INTEGER

Exemplo

Duas coleções de geometrias são armazenadas na tabela SAMPLE_GEOMCOLL. Uma é um multipolígono e a outra é um multiponto. A função ST_NumGeometries determina quantas geometrias individuais estão em cada coleção de geometrias.

```
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
```

Função ST_NumInteriorRing

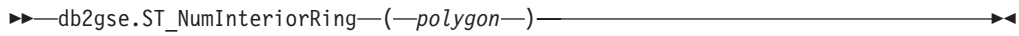
A função ST_NumInteriorRing usa um polígono como um parâmetro de entrada e retorna o número de seus anéis internos.

Se o polígono especificado for nulo ou vazio, será retornado nulo.

Se o polígono não tiver anéis internos, será retornado 0 (zero).

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

polígono
Um valor do tipo ST_Polygon que representa o polígono para o qual o número de anéis internos é retornado.

Tipo de retorno

INTEGER

Exemplo

O exemplo a seguir cria dois polígonos:

- Um com dois anéis internos
- Um sem nenhum anel interno

```
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) )
```

A função ST_NumInteriorRing é utilizada para retornar o número de anéis nas geometrias na tabela:

```
SELECT id, ST_NumInteriorRing(geometry) NUM_RINGS
FROM sample_polys
```

Resultados:

```
ID         NUM_RINGS
---------- ---------------
         1               2
         2               0
```

Função ST_NumLineStrings

A função ST_NumLineStrings usa uma sequência multilinha como um parâmetro de entrada e retorna o número de sequências de linhas que ele contém.

Se a sequência multilinha especificada for nula ou vazia, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

multilinestring
Um valor do tipo ST_MultiLineString que representa a sequência de linhas múltiplas para a qual o número de sequências de linhas é retornado.

Tipo de retorno

INTEGER

Exemplo

As sequências multilinhas são armazenadas na tabela SAMPLE_MLINES. A função ST_NumLineStrings determina quantas geometrias individuais estão em cada sequência multilinha.

```
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
```

Função ST_NumPoints

A função ST_NumPoints usa uma geometria como um parâmetro de entrada e retorna o número de pontos que foram usados para definir essa geometria. Por exemplo, se a geometria for um polígono e se foram utilizados cinco pontos para definir esse polígono, o número retornado será 5.

Se a geometria especificada for nula ou vazia, então nulo é retornado.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria para a qual o número de pontos é retornado.

Tipo de retorno

INTEGER

Exemplo

Várias geometrias são armazenadas na tabela. A função ST_NumPoints determina quantos pontos estão em cada geometria na tabela 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
```

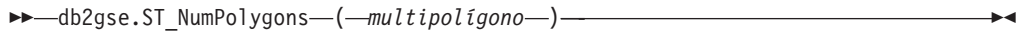
Função ST_NumPolygons

A função ST_NumPolygons usa um multipolígono como um parâmetro de entrada e retorna o número de polígonos que ele contém.

Se o multipolígono especificado for nulo ou vazio, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

multipolygon
Um valor do tipo ST_MultiPolygon que representa o multipolígono para o qual o número de polígonos é retornado.

Tipo de retorno

INTEGER

Exemplo

Os multipolígonos são armazenados na tabela SAMPLE_MPOLYS. A função ST_NumPolygons determina quantas geometrias individuais estão em 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
```

Função ST_Overlaps

A função ST_Overlaps usa duas geometrias como parâmetros de entrada. Se a intersecção das geometrias resultar em uma geometria da mesma dimensão, mas não for igual a nenhuma das geometrias especificadas, ela retornará 1. Caso contrário, retornará 0 (zero).

Se qualquer uma das duas geometrias for nula ou vazia, será retornado nulo.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria, ela será convertida para o outro sistema de referência espacial.

Sintaxe

Parâmetros

geometry1
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que será testada para sobreposição com *geometry2*.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que será testada para sobreposição com *geometry1*.

Tipo de retorno

INTEGER

Exemplos

Exemplo 1

Este exemplo ilustra a utilização de ST_Overlaps. Várias geometrias são criadas e inseridas na tabela 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) )
```

Exemplo 2

Este exemplo encontra os IDs de pontos de sobreposição.

```
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
```

Exemplo 3

Este exemplo encontra os IDs de linhas de sobreposição.

```
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
```

Exemplo 4

Este exemplo encontra os IDs de polígonos de sobreposição.

```
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
```

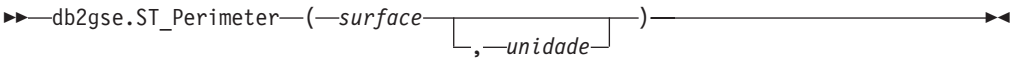
Função ST_Perimeter

A função ST_Perimeter usa uma superfície ou superfície múltipla e, opcionalmente, uma unidade como parâmetros de entrada e retorna o perímetro da superfície ou superfície múltipla, ou seja, o comprimento de seu limite, medido nas unidades padrão ou especificadas.

Se a superfície ou multisuperfície especificada for nula ou vazia, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

surface

Um valor do tipo ST_Surface, ST_MultiSurface ou um de seus subtipos para o qual o perímetro é retornado.

unit Um valor VARCHAR(128) que identifica as unidades nas quais o perímetro é medido. As unidades de medida suportadas estão listadas na exibição do catálogo DB2GSE.ST_UNITS_OF_MEASURE.

Se o parâmetro *unit* for omitido, as seguintes regras serão utilizadas para determinar a unidade na qual o perímetro é medido:

- Se *surface* estiver em um sistema de coordenadas projetadas ou geocêntricas, a unidade linear associada a este sistema de coordenadas será a padrão.
- Se *surface* estiver em um sistema de coordenadas geográficas, a unidade angular associada a este sistema de coordenadas será o padrão.

Restrições para conversões de unidades: Será retornado um erro (SQLSTATE 38SU4) se ocorrer qualquer uma das condições a seguir:

- A geometria está em um sistema de coordenadas não especificado e o parâmetro *unit* é especificado.
- A geometria está em um sistema de coordenadas projetadas e uma unidade angular é especificada.
- A geometria está em um sistema de coordenadas geográficas e uma unidade linear é especificada.

Tipo de retorno

DOUBLE

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_Perimeter. É criado um sistema de referência espacial com um ID 4000 utilizando uma chamada para db2se, e é criado um polígono nesse sistema de referência 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
```

A tabela SAMPLE_POLYS é criada para conter uma geometria com um perímetro 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))
```

Exemplo 2

Este exemplo lista o ID e o perímetro do polígono.

```
SELECT id, ST_Perimeter (geometry) AS PERIMETER
  FROM sample_polys
```

Resultados:

```
ID          PERIMETER
----------- ------------------------
          1 +1.80000000000000E+001
```

Exemplo 3

Este exemplo lista o ID e o perímetro do polígono com o perímetro medido em metros.

```
SELECT id, ST_Perimeter (geometry, 'METER') AS PERIMETER_METER
  FROM sample_polys
```

Resultados:

```
ID          PERIMETER_METER
----------- ----------------------
          1 +5.48641097282195E+000
```

Função ST_PerpPoints

A função ST_PerpPoints usa uma curva ou multicurva e um ponto como parâmetros de entrada e retorna a projeção perpendicular do ponto especificado na curva ou multicurva.

É retornado o ponto com a menor distância entre o ponto especificado e o ponto perpendicular. Se dois ou mais desses pontos projetados perpendiculares forem equidistantes do ponto especificado, serão todos retornados. Se nenhum ponto perpendicular puder ser construído, será retornado um ponto vazio.

Se a curva ou multicurva especificada tiver coordenadas Z ou M, as coordenadas Z ou M dos pontos resultantes serão calculadas por interpolação na curva ou multicurva especificada.

Se a curva ou ponto especificado for vazio, será retornado um ponto vazio. Se a curva ou ponto for nulo, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetros

curve Um valor do tipo ST_Curve, ST_MultiCurve ou um de seus subtipos que representa a curva ou multicurva na qual a projeção perpendicular do *ponto* é retornada.

ponto Um valor do tipo ST_Point que representa o ponto perpendicular projetado em *curva*.

Tipo de retorno

db2gse.ST_MultiPoint

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_PerpPoints para localizar pontos que são perpendiculares à sequência de linhas armazenada na seguinte tabela. A função ST_LineString é utilizada na instrução INSERT para criar a sequência de linhas.

```
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) )
```

Exemplo 2

Este exemplo localiza a projeção perpendicular na sequência de linhas de um ponto com coordenadas (5, 0). A função ST_AsText é utilizada para converter o valor retornado (um multiponto) em sua representação de texto reconhecida.

```
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)
```

Exemplo 3

Este exemplo localiza as projeções perpendiculares na sequência de linhas de um ponto com coordenadas (5, 5). Neste caso, existem três pontos na sequência de linhas que estão equidistantes da localização especificada. Portanto, é retornado um multiponto que consiste nos três pontos.

```
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)
```

Exemplo 4

Este exemplo localiza as projeções perpendiculares na sequência de linhas de um ponto com coordenadas (5, 10). Neste caso, existem três diferentes pontos perpendiculares que podem ser encontrados. No entanto, a função ST_PerpPoints somente retorna os pontos que estão mais próximos do ponto especificado. Assim, é retornado um multiponto que consiste apenas nos dois pontos mais próximos. O terceiro ponto não é incluído.

```
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 )
```

Exemplo 5

Este exemplo localiza a projeção perpendicular na sequência de linhas de um ponto com coordenadas (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 )
```

Exemplo 6

Neste exemplo, o ponto especificado com coordenadas (15 15) não possui nenhuma projeção perpendicular na sequência de linhas. Portanto, é retornada uma geometria vazia.

```
SELECT CAST(ST_AsText(ST_PerpPoints(line,ST_Point(15,15)))
  AS VARCHAR(80) ) PERP
  FROM sample_lines
```

Resultados:

```
PERP
--------------------------------------------
MULTIPOINT EMPTY
```

Função ST_Point

A função ST_Point constrói um ponto a partir das informações de coordenadas ou um ponto resultante de uma representação.

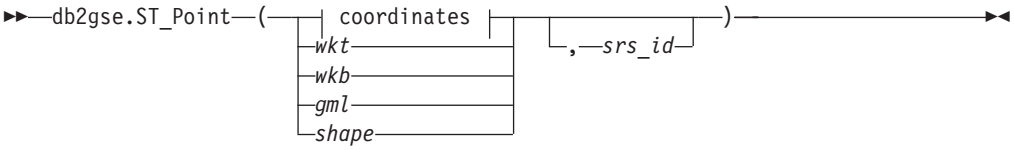
ST_Point constrói um ponto a partir de um dos seguintes conjuntos de entrada:

- Somente coordenadas X e Y
- Coordenadas X, Y e Z
- Coordenadas X, Y, Z e M
- Uma representação de texto reconhecida
- Uma representação binária reconhecida
- Uma representação de formatos
- Uma representação na linguagem de marcação geográfica (GML)

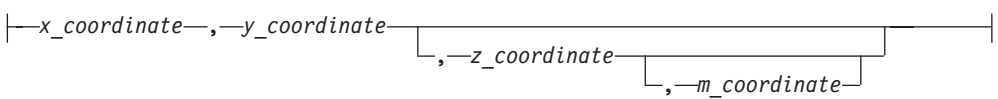
Um identificador do sistema de referência espacial opcional pode ser especificado para indicar o sistema de referência espacial no qual o ponto resultante está localizado.

Se o ponto for construído a partir de coordenadas e se a coordenada X ou Y for nula, ocorrerá uma condição de exceção (SQLSTATE 38SUP). Se a coordenada Z ou M for nula, o ponto resultante não terá uma coordenada Z ou M, respectivamente. Se o ponto for construído a partir de sua representação de texto reconhecida, de sua representação binária reconhecida, de sua representação de formatos ou de sua representação GML, e se a representação for nula, será retornado nulo.

Sintaxe



coordenadas:



Parâmetros

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida do ponto resultante.

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida do ponto resultante.

gml Um valor do tipo CLOB(2G) que representa o ponto resultante utilizando a linguagem de marcação geográfica.

shape Um valor do tipo BLOB(2G) que representa a representação de formatos do ponto resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do ponto resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

x_coordinate
Um valor do tipo DOUBLE que especifica a coordenada X para o ponto resultante.

y_coordinate
Um valor do tipo DOUBLE que especifica a coordenada Y para o ponto resultante.

z_coordinate
Um valor do tipo DOUBLE que especifica a coordenada Z para o ponto resultante.

Se o parâmetro *z_coordinate* for omitido, o ponto resultante não terá uma coordenada Z. O resultado de ST_Is3D será 0 (zero) para tal ponto.

m_coordinate
Um valor do tipo DOUBLE que especifica a coordenada M para o ponto resultante.

Se o parâmetro *m_coordinate* for omitido, o ponto resultante não terá uma medida. O resultado de ST_IsMeasured será 0 (zero) para tal ponto.

Tipo de retorno

db2gse.ST_Point

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Exemplo 1

Este exemplo ilustra como ST_Point pode ser utilizado para criar e inserir pontos. O primeiro ponto é criado utilizando um conjunto de coordenadas X e Y. O segundo ponto é criado utilizando sua representação de texto reconhecida. Ambos os pontos são geometrias no sistema de referência 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) )
```

A instrução SELECT a seguir retorna os pontos que foram registrados na tabela:

```
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)
```

Exemplo 2

Este exemplo insere um registro na tabela SAMPLE_POINTS com o ID 1103 e um valor de ponto com uma coordenada X de 120, uma coordenada Y de 358, uma coordenada M de 34, mas nenhuma 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)
```

Exemplo 3

Este exemplo insere uma linha na tabela SAMPLE_POINTS com ID 1104 e um valor de ponto com uma coordenada X de 1003, uma coordenada Y de 9876, uma coordenada Z de 20 e um sistema de referência espacial 0, utilizando a linguagem de marcação geográfica para sua representação.

```
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:

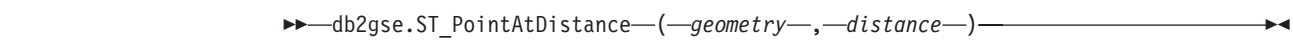
```
ID         POINTS
 ---------- -------------------------------------------------
       1104 POINT Z ( 1003.000000 9876.000000 20.00000000)
```

função ST_PointAtDistance

A função ST_PointAtDistance usa uma geometria de curva ou de multicurva e uma distância como parâmetros de entrada e retorna a geometria de ponto na distância especificada na geometria de curva.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Curve ou ST_MultiCurve ou um de seus subtipos que representa a geometria a ser processada.

distance
Um valor do tipo DOUBLE que é a distância ao longo da geometria para localizar o ponto.

Tipo de retorno

db2gse.ST_Point

Exemplo

A seguinte instrução SQL cria a tabela SAMPLE_GEOMETRIES com duas colunas. A coluna do ID identifica exclusivamente cada linha. A coluna GEOMETRY ST_LineString armazena geometrias de amostra.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

A seguinte instrução SQL insere duas linhas na tabela SAMPLE_GEOMETRIES.

```
INSERT INTO sample_geometries(id, geometry)
VALUES
 (1,ST_LineString('LINESTRING ZM(0 0 0 0, 10 100 1000 10000)',1)),
 (2,ST_LineString('LINESTRING ZM(10 100 1000 10000, 0 0 0 0)',1))
```

A seguinte instrução SELECT e o conjunto de resultados correspondente mostram como usar a função ST_PointAtDistance para localizar pontos a uma distância de 15 unidades coordenadas do início da sequência de linhas.

```
SELECT ID, VARCHAR(ST_AsText(ST_PointAtDistance(geometry, 15)), 50) AS POINTAT
FROM   sample_geometries

ID          POINTAT
----------- --------------------------------------------------
          1 POINT ZM(1.492556 14.925558 149 1493)
          2 POINT ZM(8.507444 85.074442 851 8507)

  2 record(s) selected.
```

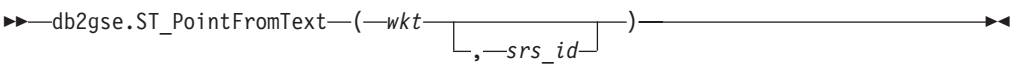
Função ST_PointFromText

A função ST_PointFromText usa uma representação well-known text de um ponto e, opcionalmente, um identificador de sistema de referência espacial como parâmetros de entrada e retorna o ponto correspondente.

Se a representação de texto reconhecida for nula, então nulo é retornado.

A função recomendada para alcançar o mesmo resultado é a função ST_Point. Ela é recomendada por sua flexibilidade: ST_Point utiliza formatos adicionais de entrada, além da representação de texto reconhecida.

Sintaxe



Parâmetros

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida do ponto resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do ponto resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_Point

Exemplo

Este exemplo ilustra como ST_PointFromText pode ser utilizado para criar e inserir um ponto a partir de sua representação de texto reconhecida. O registro inserido tem ID = 1110 e a geometria é um ponto no sistema de referência espacial 1. O ponto está na representação de texto reconhecida de um ponto. As coordenadas X e Y para esta geometria são: (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) )
```

A instrução SELECT a seguir retorna o polígono que foi registrado na tabela:

```
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)
```

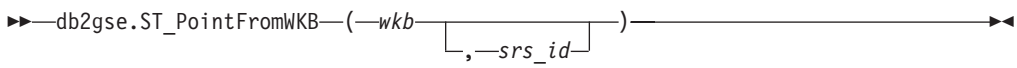
Função ST_PointFromWKB

A função ST_PointFromWKB usa uma representação binária reconhecida de um ponto e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna o ponto correspondente.

Se representação binária reconhecida especificada for nula, então nulo é retornado.

A função recomendada para alcançar o mesmo resultado é a função ST_Point. Ela é recomendada por sua flexibilidade: ST_Point utiliza formatos adicionais de entrada, além da representação binária reconhecida.

Sintaxe



Parâmetros

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida do ponto resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do ponto resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado implicitamente.

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_Point

Exemplo

Este exemplo ilustra como ST_PointFromWKB pode ser utilizado para criar um ponto a partir de sua representação binária reconhecida. As geometrias são pontos no sistema de referência espacial 1. Neste exemplo, os pontos são armazenados na coluna GEOMETRY da tabela SAMPLE_POLYS e depois a coluna WKB é atualizada com suas representações binárias reconhecidas (utilizando a função ST_AsBinary). Por último, a função ST_PointFromWKB é utilizada para retornar os pontos da coluna WKB.

A tabela SAMPLE_POINTS tem uma coluna GEOMETRY, em que os pontos são armazenados e uma coluna WKB, em que as representações binárias reconhecidas dos pontos são armazenadas.

```
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
```

Na instrução SELECT a seguir, a função ST_PointFromWKB é utilizada para recuperar os pontos da coluna 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)
```

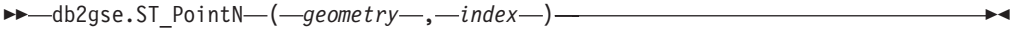
Função ST_PointN

A função ST_PointN usa uma sequência de linhas ou um multiponto e um índice como parâmetros de entrada e retorna esse ponto na sequência de linhas ou multiponto identificado pelo índice. O ponto resultante é representado no sistema de referência espacial da sequência de linhas ou multiponto especificado.

Se a sequência de linhas ou o multiponto for nulo ou vazio, será retornado nulo. Se o índice for menor do que 1 ou maior do que o número de pontos na sequência de linhas ou multiponto, será retornado nulo e também uma condição de aviso (SQLSTATE 01HS2).

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

geometria
Um valor do tipo ST_LineString ou ST_MultiPoint que representa a geometria a partir da qual o ponto identificado por *index* é retornado.

índice Um valor do tipo INTEGER que identifica o *ú*ltimo ponto que será retornado de *geometry*.

Tipo de retorno

db2gse.ST_Point

Exemplo

O exemplo a seguir ilustra a utilização 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)
```

Função ST_PointOnSurface

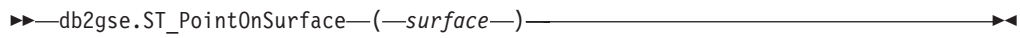
A função ST_PointOnSurface usa uma superfície ou superfície múltipla como um parâmetro de entrada e retorna um ponto que tem garantia de estar no interior da superfície ou superfície múltipla. Este ponto é o paracentróide da superfície.

O ponto resultante é representado no sistema de referência espacial da superfície ou multisuperfície especificada.

Se a superfície ou multisuperfície especificada for nula ou vazia, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

surface
Um valor do tipo ST_Surface, ST_MultiSurface ou um de seus subtipos que representa a geometria para a qual é retornado um ponto na superfície.

Tipo de retorno

db2gse.ST_Point

Exemplo

No exemplo a seguir, são criados dois polígonos e ST_PointOnSurface é utilizado. Um dos polígonos tem um orifício em seu centro. Os pontos retornados estão na superfície dos polígonos. Eles não estão necessariamente no centro dos 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)
```

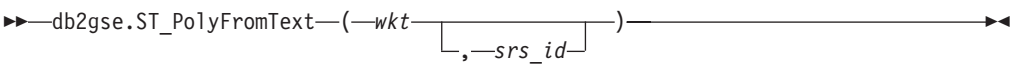
Função ST_PolyFromText

A função ST_PolyFromText usa uma representação de texto reconhecida de um polígono e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna o polígono correspondente.

Se a representação de texto reconhecida for nula, então nulo é retornado.

A função recomendada para alcançar o mesmo resultado é a função ST_Polygon. Ela é recomendada por sua flexibilidade: ST_Polygon utiliza formatos adicionais de entrada, além da representação de texto reconhecida.

Sintaxe



Parâmetros

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida do polígono resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do polígono resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_Polygon

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_PolyFromText pode ser utilizado para criar e inserir um polígono a partir de sua representação de texto reconhecida. O registro inserido tem ID = 1110 e a geometria é um polígono no sistema de referência espacial 1. O polígono está na representação de texto reconhecida de um polígono. As coordenadas X e Y para esta geometria são: (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) )
```

A instrução SELECT a seguir retorna o polígono que foi registrado na tabela:

```
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))
```

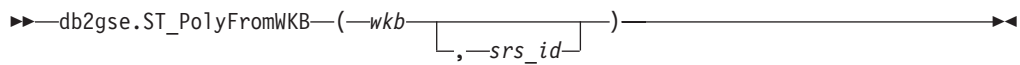
Função ST_PolyFromWKB

A função ST_PolyFromWKB usa uma representação de binário reconhecido de um polígono e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada e retorna o polígono correspondente.

Se representação binária reconhecida especificada for nula, então nulo é retornado.

A função recomendada para alcançar o mesmo resultado é a função ST_Polygon. Ela é recomendada por sua flexibilidade: ST_Polygon utiliza formatos adicionais de entrada, além da representação binária reconhecida.

Sintaxe



Parâmetros

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida do polígono resultante.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do polígono resultante.

Se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) será utilizado.

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_Polygon

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_PolyFromWKB pode ser utilizado para criar um polígono a partir de sua representação binária reconhecida. A geometria é um polígono no sistema de referência espacial 1. Neste exemplo, o polígono é armazenado com ID = 1115 na coluna GEOMETRY da tabela SAMPLE_POLYS e, em seguida, a coluna WKB é atualizada com sua representação binária reconhecida (utilizando a função ST_AsBinary). Por último, a função ST_PolyFromWKB é utilizada para retornar o multipolígono da coluna WKB. As coordenadas X e Y para esta geometria são: (50, 20) (50, 40) (70, 30).

A tabela SAMPLE_POLYS tem uma coluna GEOMETRY, em que o polígono é armazenado e uma coluna WKB, em que a representação binária reconhecida do polígono é armazenada.

```
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
```

Na instrução SELECT a seguir, a função ST_PolyFromWKB é utilizada para recuperar o polígono da coluna 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))
```

Função ST_Polygon

ST_Polygon constrói um polígono a partir de uma entrada especificada.

A entrada pode ser especificada em um dos seguintes formatos:

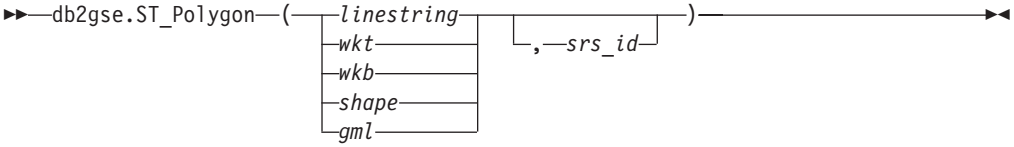
- Uma sequência de linhas fechada que define o anel externo do polígono resultante
- Uma representação de texto reconhecida
- Uma representação binária reconhecida
- Uma representação de formatos
- Uma representação na linguagem de marcação geográfica (GML)

Um identificador do sistema de referência espacial opcional pode ser especificado para identificar o sistema de referência espacial no qual o polígono resultante está localizado.

Se o polígono for construído a partir de uma sequência de linhas e a sequência de linhas especificada for nula, será retornado nulo. Se a sequência de linhas especificada estiver vazia, será retornado um polígono vazio. Se o polígono for construído a partir de sua representação de texto reconhecida, de sua representação binária reconhecida, de sua representação de formatos ou de sua representação GML, e se a representação for nula, será retornado nulo.

Esta função também pode ser chamada como um método somente nos seguintes casos: ST_Polygon(*linestring*) e ST_Polygon(*linestring*, *srs_id*).

Sintaxe



Parâmetros

sequência-de-linhas
Um valor do tipo ST_LineString que representa a sequência de linhas que define o anel externo do limite externo. Se *linestring* não estiver fechado e for simples, ocorrerá uma condição de exceção (SQLSTATE 38SSL).

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida do polígono resultante.

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida do polígono resultante.

shape Um valor do tipo BLOB(2G) que representa a representação de formatos do polígono resultante.

gml Um valor do tipo CLOB(2G) que representa o polígono resultante utilizando a linguagem de marcação geográfica.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial do polígono resultante.

Se o polígono for construído a partir de um parâmetro *linestring* especificado e o parâmetro *srs_id* for omitido, o sistema de referência espacial de *linestring* será utilizado implicitamente. Caso contrário, se o parâmetro *srs_id* for omitido, o sistema de referência espacial com o identificador numérico 0 (zero) é utilizado.

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_Polygon

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_Polygon pode ser utilizado para criar e inserir polígonos. Três polígonos são criados e inseridos. Todos eles são geometrias no sistema de referência espacial 1.

- O primeiro polígono é criado a partir de um anel (uma sequência de linhas fechada e simples). As coordenadas X e Y para este polígono são: (10, 20) (10, 40) (20, 30).
- O segundo polígono é criado utilizando sua representação de texto reconhecida. As coordenadas X e Y para este polígono são: (110, 120) (110, 140) (120, 130).
- O terceiro polígono é um polígono circular. Um polígono circular consiste em um polígono interno e um externo. Este polígono circular é criado utilizando sua representação de texto reconhecida. As coordenadas X e Y para o polígono externo são: (110, 120) (110, 140) (130, 140) (130, 120) (110, 120). As coordenadas X e Y para o polígono interno são: (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))
```

A instrução SELECT a seguir retorna os polígonos que foram registrados na tabela:

```
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))
```

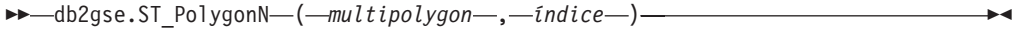
Função ST_PolygonN

A função ST_PolygonN usa um multipolígono e um índice como parâmetros de entrada e retorna o polígono identificado pelo índice. O polígono resultante é representado no sistema de referência espacial do multipolígono especificado.

Se o multipolígono especificado for nulo ou vazio, ou se o índice for menor do que 1 ou maior do que o número de polígonos, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

multipolygon
Um valor do tipo ST_MultiPolygon que representa o multipolígono a partir do qual o polígono identificado por *index* é retornado.

índice Um valor do tipo INTEGER que identifica o *ú*ltimo polígono que será retornado de *multipolygon*.

Tipo de retorno

db2gse.ST_Polygon

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento em seus resultados variará, de acordo com sua exibição on-line

Este exemplo ilustra a utilização 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))
```

Função ST_Relate

A função ST_Relate usa duas geometrias e uma matriz Dimensionally Extended 9 Intersection Model (DE-9IM) como parâmetros de entrada. Se as geometrias especificadas atenderem às condições especificadas pela matriz, ela retornará 1. Caso contrário, retornará 0.

Se qualquer uma das geometrias especificadas for nula ou vazia, será retornado nulo.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria, ela será convertida para o outro sistema de referência espacial.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetros

geometry1
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que será testada em *geometry2*.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que será testada contra *geometry1*.

matrix Um valor do tipo CHAR(9) que representa a matriz DE-9IM que será utilizada para o teste de *geometry1* e *geometry2*.

Tipo de retorno

INTEGER

Exemplo

O código a seguir cria dois polígonos separados. Em seguida, a função ST_Relate é utilizada para determinar vários relacionamentos entre os dois polígonos. Por exemplo, se os dois polígonos são sobrepostos.

```
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
```

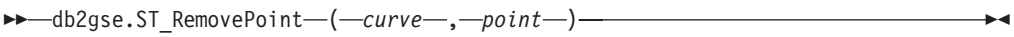
Função ST_RemovePoint

A função ST_RemovePoint usa uma curva e um ponto como parâmetros de entrada e retorna a curva especificada com todos os pontos iguais ao ponto especificado removido dela. Se a curva especificada tiver coordenadas Z ou M, o ponto também deverá ter coordenadas Z ou M. A geometria resultante é representada no sistema de referência espacial da geometria especificada.

Se a curva especificada for vazia, então um curva vazia será retornada. Se a curva especificada for nula, ou se o ponto especificado for nulo ou vazio, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

curve Um valor do tipo ST_Curve ou um de seus subtipos que representa a curva a partir da qual *point* é removido.

ponto Um valor do tipo ST_Point que identifica os pontos que são removidos de *curve*.

Tipo de retorno

db2gse.ST_Curve

Exemplos

Exemplo 1

No exemplo a seguir, duas sequências de linhas são incluídas na tabela SAMPLE_LINES. Estas sequências de linhas são usadas nos exemplos a seguir.

```
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))
```

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Exemplo 2

O exemplo a seguir remove o ponto (5, 5) da sequência de linhas que tem ID = 1. Este ponto ocorre duas vezes na sequência de linhas. Portanto, as duas ocorrências são removidas.

```
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)
```

Exemplo 3

O exemplo a seguir remove o ponto (5, 5, 5) da sequência de linhas que tem ID = 2. Este ponto ocorre somente uma vez, portanto, somente essa ocorrência é removida.

```
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)
```

Função ST_SrsId ou ST_SRID

A função ST_SrsId ou ST_SRID usa uma geometria e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada.

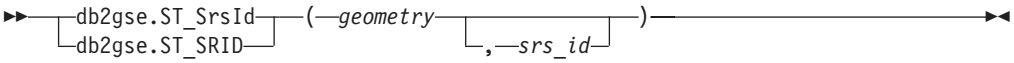
O que ela retorna depende de quais parâmetros de entrada foram especificados:

- Se o identificador do sistema de referência espacial for especificado, será retornada uma geometria com seu sistema de referência espacial alterado para o sistema de referência espacial especificado. Nenhuma transformação da geometria é executada.
- Se nenhum identificador do sistema de referência espacial for especificado como parâmetro de entrada, será retornado o identificador do sistema de referência espacial atual da geometria especificada.

Se a geometria indicada for nula, será retornado nulo.

Estas funções também podem ser chamadas como métodos.

Sintaxe



Parâmetros

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria para a qual o identificador do sistema de referência espacial será definido ou retornado.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial que será utilizado para a geometria resultante.

Atenção: Se este parâmetro for especificado, a geometria não será transformada, mas será retornada com seu sistema de referência espacial alterado para o sistema de referência espacial especificado. Como resultado da alteração para o novo sistema de referência espacial, os dados poderão estar danificados. Para transformações, utilize ST_Transform.

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

Tipos de retornos

- INTEGER, se um srs_id não for especificado
- db2gse.ST_Geometry, se um srs_id for especificado

Exemplo

São criados dois pontos em dois sistemas de referência espacial diferentes. O ID do sistema de referência espacial que está associado a cada ponto pode ser encontrado utilizando a função 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
```

Função ST_SrsId ou ST_SRID

A função ST_SrsId ou ST_SRID usa uma geometria e, opcionalmente, um identificador do sistema de referência espacial como parâmetros de entrada.

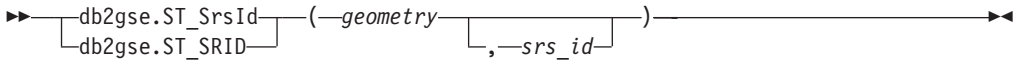
O que ela retorna depende de quais parâmetros de entrada foram especificados:

- Se o identificador do sistema de referência espacial for especificado, será retornada uma geometria com seu sistema de referência espacial alterado para o sistema de referência espacial especificado. Nenhuma transformação da geometria é executada.
- Se nenhum identificador do sistema de referência espacial for especificado como parâmetro de entrada, será retornado o identificador do sistema de referência espacial atual da geometria especificada.

Se a geometria indicada for nula, será retornado nulo.

Estas funções também podem ser chamadas como métodos.

Sintaxe



Parâmetros

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria para a qual o identificador do sistema de referência espacial será definido ou retornado.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial que será utilizado para a geometria resultante.

Atenção: Se este parâmetro for especificado, a geometria não será transformada, mas será retornada com seu sistema de referência espacial alterado para o sistema de referência espacial especificado. Como resultado da alteração para o novo sistema de referência espacial, os dados poderão estar danificados. Para transformações, utilize ST_Transform.

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

Tipos de retornos

- INTEGER, se um srs_id não for especificado
- db2gse.ST_Geometry, se um srs_id for especificado

Exemplo

São criados dois pontos em dois sistemas de referência espacial diferentes. O ID do sistema de referência espacial que está associado a cada ponto pode ser encontrado utilizando a função 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
```

Função ST_SrsName

A função ST_SrsName usa uma geometria como um parâmetro de entrada e retorna o nome do sistema de referência espacial no qual a geometria especificada é representada.

Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria para a qual o nome do sistema de referência espacial é retornado.

Tipo de retorno

VARCHAR(128)

Exemplo

São criados dois pontos em sistemas de referência espacial diferentes. A função ST_SrsName é utilizada para encontrar o nome do sistema de referência espacial que está associado a cada ponto.

```
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
```

Função ST_StartPoint

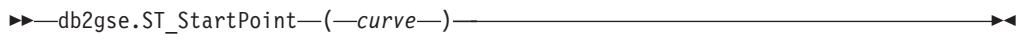
A função ST_StartPoint usa uma curva como um parâmetro de entrada e retorna o ponto que é o primeiro ponto da curva.

O ponto resultante é representado no sistema de referência espacial da curva especificada. Este resultado é equivalente à chamada de função ST_PointN(*curve*, 1)

Se a curva especificada for nula ou vazia, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

curve Um valor do tipo ST_Curve ou um de seus subtipos que representa a geometria a partir da qual o primeiro ponto é retornado.

Tipo de retorno

db2gse.ST_Point

Exemplo

No exemplo a seguir, duas sequências de linhas são incluídas na tabela SAMPLE_LINES. A primeira é uma sequência de linhas com coordenadas X e Y. A

segunda é uma sequência de linhas com coordenadas X, Y e Z. A função ST_StartPoint é utilizada para retornar o primeiro ponto em cada sequência de linhas.

```
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)
```

Função ST_SymDifference

A função ST_SymDifference usa duas geometrias como parâmetros de entrada e retorna a geometria que é diferença simétrica das duas geometrias. A diferença simétrica é a parte sem intersecção das duas geometrias determinadas.

A geometria resultante é representada no sistema de referência espacial da primeira geometria. A dimensão da geometria retornada é igual à das geometrias de entrada. As duas geometrias devem ter a mesma dimensão.

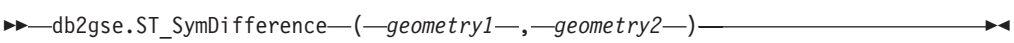
Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria e usar o mesmo datum subjacente, ela será convertida no outro sistema de referência espacial.

Se as geometrias forem iguais, será retornada uma geometria vazia do tipo ST_Point. Se qualquer uma das geometrias for nula, será retornado nulo.

A geometria resultante é representada no tipo espacial mais apropriado. Se puder ser representada como um ponto, sequência de linhas ou polígono, será utilizado um desses tipos. Caso contrário, será utilizado o tipo multiponto, sequência multilinha ou multipolígono.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

geometry1
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a primeira geometria para calcular a diferença simétrica com *geometry2*.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a segunda geometria para calcular a diferença simétrica com *geometry1*.

Tipo de retorno

db2gse.ST_Geometry

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_SymDifference. As geometrias são armazenadas na tabela 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));
```

Nos exemplos a seguir, os resultados foram reformatados para possibilitar a leitura. Seus resultados variarão, de acordo com sua exibição.

Exemplo 2

Este exemplo utiliza ST_SymDifference para retornar a diferença simétrica de dois polígonos separados na tabela 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)))
```

Exemplo 3

Este exemplo utiliza ST_SymDifference para retornar a diferença simétrica de dois polígonos que fazem interseção na tabela 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)))
```

Exemplo 4

Este exemplo utiliza ST_SymDifference para retornar a diferença simétrica de duas sequências de linhas que fazem interseção na tabela 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))
```

Função ST_ToGeomColl

A função ST_ToGeomColl usa uma geometria como um parâmetro de entrada e converte-a em uma coleção de geometrias. A coleção de geometria resultante é representada no sistema de referência espacial da geometria especificada.

Se a geometria especificada for vazia, ela poderá ser de qualquer tipo. No entanto, ela será convertida em ST_Multipoint, ST_MultiLineString ou ST_MultiPolygon conforme apropriado.

Se a geometria especificada não for vazia, ela deverá ser do tipo ST_Point, ST_LineString ou ST_Polygon. Elas serão convertidas em ST_Multipoint, ST_MultiLineString ou ST_MultiPolygon, respectivamente.

Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que é convertida para uma coleção de geometria.

Tipo de retorno

db2gse.ST_GeomCollection

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento em seus resultados variará, de acordo com sua exibição on-line

Este exemplo ilustra a utilização da função 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))
```

Na instrução SELECT a seguir, a função ST_ToGeomColl é utilizada para retornar geometrias como seus subtipos de coleção de geometria correspondentes.

```
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)
```

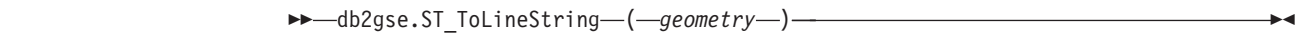
Função ST_ToLineString

A função ST_ToLineString usa uma geometria como um parâmetro de entrada e converte-a em uma sequência de linhas. A sequência de linhas resultante é representada no sistema de referência espacial da geometria especificada.

A geometria especificada deve ser vazia ou uma sequência de linhas. Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria

Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que é convertida para uma sequência de linhas.

Uma geometria pode ser convertida em uma sequência de linhas se for vazia ou uma sequência de linhas. Se a conversão não puder ser executada, ocorrerá uma condição de exceção (SQLSTATE 38SUD).

Tipo de retorno

db2gse.ST_LineString

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento em seus resultados variará, de acordo com sua exibição on-line

Este exemplo ilustra a utilização da função 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) )
```

Na instrução SELECT a seguir, a função ST_ToLineString é utilizada para retornar sequências de linhas convertidas em ST_LineString a partir do tipo estático de 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
```

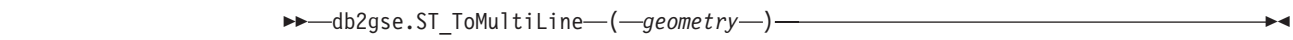
Função ST_ToMultiLine

A função ST_ToMultiLine usa uma geometria como um parâmetro de entrada e converte-a em uma sequência multilinha. A sequência de múltiplas linhas resultante é representada no sistema de referência espacial da geometria especificada.

A geometria especificada deve ser vazia, uma sequência multilinha ou uma sequência de linhas. Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria

Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que é convertida para uma sequência de linhas múltiplas.

Uma geometria pode ser convertida em uma sequência multilinha se for vazia, uma sequência de linhas ou uma sequência multilinha. Se a conversão não puder ser executada, ocorrerá uma condição de exceção (SQLSTATE 38SUD).

Tipo de retorno

db2gse.ST_MultiLineString

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento em seus resultados variará, de acordo com sua exibição on-line

Este exemplo ilustra a utilização da função 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) )
```

Na instrução SELECT a seguir, a função ST_ToMultiLine é utilizada para retornar sequências multilinha convertidas em ST_MultiLineString a partir do tipo estático de 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
```

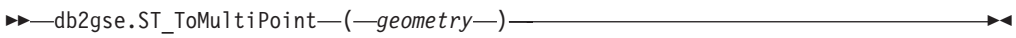
Função ST_ToMultiPoint

A função ST_ToMultiPoint usa uma geometria como um parâmetro de entrada e converte-a em um multiponto. O multiponto resultante é representado no sistema de referência espacial da geometria especificada.

A geometria especificada deve ser vazia, um ponto ou um multiponto. Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que é convertida em um multiponto.

Uma geometria pode ser convertida em um multiponto se for vazia, um ponto ou um multiponto. Se a conversão não puder ser executada, ocorrerá uma condição de exceção (SQLSTATE 38SUD).

Tipo de retorno

db2gse.ST_MultiPoint

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento em seus resultados variará, de acordo com sua exibição on-line

Este exemplo ilustra a utilização da função 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) )
```

Na instrução SELECT a seguir, a função ST_ToMultiPoint é utilizada para retornar multipontos convertidos em ST_MultiPoint a partir do tipo estático de 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
```

Função ST_ToMultiPolygon

A função ST_ToMultiPolygon usa uma geometria como um parâmetro de entrada e converte-a em um multipolígono. O multipolígono resultante é representado no sistema de referência espacial da geometria especificada.

A geometria especificada deve ser vazia, um polígono ou um multipolígono. Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria

Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que é convertida em um multipolígono.

Uma geometria pode ser convertida em um multipolígono se for vazia, polígono ou multipolígono. Se a conversão não puder ser executada, ocorrerá uma condição de exceção (SQLSTATE 38SUD).

Tipo de retorno

db2gse.ST_MultiPolygon

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento em seus resultados variará, de acordo com sua exibição on-line

Este exemplo cria várias geometrias e depois utiliza ST_ToMultiPolygon para retornar 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))
```

Na instrução SELECT a seguir, a função ST_ToMultiPolygon é utilizada para retornar multipolígonos convertidos em ST_MultiPolygon a partir do tipo estático de 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
```

Função ST_ToPoint

A função ST_ToPoint usa uma geometria como um parâmetro de entrada e converte-a em um ponto. O ponto resultante é representado no sistema de referência espacial da geometria especificada.

A geometria especificada deve ser vazia ou um ponto. Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetro

geometria

Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que é convertida em um ponto.

Uma geometria pode ser convertida em um ponto se for vazia ou um ponto. Se a conversão não puder ser executada, ocorrerá uma condição de exceção (SQLSTATE 38SUD).

Tipo de retorno

db2gse.ST_Point

Exemplo

Este exemplo cria três geometrias em SAMPLE_GEOMETRIES e converte cada uma em um ponto.

```
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) )
```

Na instrução SELECT a seguir, a função ST_ToPoint é utilizada para retornar pontos convertidos em ST_Point a partir do tipo estático de 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
```

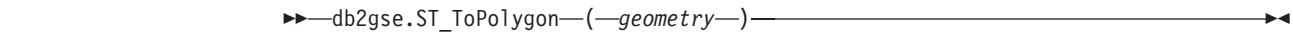
Função ST_ToPolygon

ST_ToPolygon usa uma geometria como um parâmetro de entrada e converte-a em um polígono. O polígono resultante é representado no sistema de referência espacial da geometria especificada.

A geometria especificada deve ser vazia ou um polígono. Se a geometria indicada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetro

geometria

Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que é convertida em um polígono.

Uma geometria pode ser convertida em um polígono se for vazia ou um polígono. Se a conversão não puder ser executada, ocorrerá uma condição de exceção (SQLSTATE 38SUD).

Tipo de retorno

db2gse.ST_Polygon

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento em seus resultados variará, de acordo com sua exibição on-line

Este exemplo cria três geometrias em SAMPLE_GEOMETRIES e converte cada uma em um 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) )
```

Na instrução SELECT a seguir, a função ST_ToPolygon é utilizada para retornar polígonos convertidos em ST_Polygon a partir do tipo estático de 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
```

Função ST_Touches

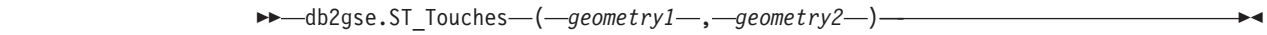
A função ST_Touches usa duas geometrias como parâmetros de entrada e retorna 1 se as geometrias especificadas espacialmente se tocarem. Caso contrário, será retornado 0 (zero).

Duas geometrias se cruzam se os interiores das duas não fizerem interseção, mas o limite de uma das geometrias fizer interseção com o limite ou o interior da outra geometria.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria, ela será convertida para o outro sistema de referência espacial.

Se as duas geometrias especificadas forem pontos ou multipontos, ou se qualquer uma das geometrias especificadas for nula ou vazia, será retornado nulo.

Sintaxe



Parâmetros

geometry1
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que será testada para cruzar com *geometry2*.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que será testada para cruzar com *geometry1*.

Tipo de retorno

INTEGER

Exemplo

Várias geometrias são incluídas na tabela SAMPLE_GEOMS. A função ST_Touches é utilizada para determinar quais geometrias se cruzam.

```
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
```

Função ST_Transform

A função ST_Transform usa uma geometria e um identificador do sistema de referência espacial como parâmetros de entrada e transforma a geometria a ser representada no sistema de referência espacial especificado. As projeções e conversões entre diferentes sistemas de coordenadas são executadas e as coordenadas das geometrias são ajustadas de acordo.

A geometria somente pode ser convertida no sistema de referência espacial especificado se o sistema de referência espacial atual da geometria estiver baseado no mesmo sistema de coordenadas geográficas que o sistema de referência espacial especificado. Se nem o sistema de referência espacial atual nem o sistema de referência espacial especificado da geometria estiver baseado em um sistema de coordenadas projetadas, será executada uma projeção reversa para determinar o sistema de coordenadas geográficas que suporta o projetado.

Se a geometria especificada for nula, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe

Parâmetros

geometria
Um valor do tipo ST_Geometry ou um de seus subtipos que representa a geometria que é transformada no sistema de referência espacial identificado por *srs_id*.

srs_id Um valor do tipo INTEGER que identifica o sistema de referência espacial da geometria resultante.

Se a transformação no sistema de referência espacial especificado não puder ser executada porque o sistema de referência espacial atual da *geometria* não é compatível com o sistema de referência espacial identificado por *srs_id*, ocorrerá uma condição de exceção (SQLSTATE 38SUC).

Se *srs_id* não identificar um sistema de referência espacial listado na exibição do catálogo DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS, ocorrerá uma condição de exceção (SQLSTATE 38SU1).

Tipo de retorno

db2gse.ST_Geometry

Exemplos

Exemplo 1

O exemplo a seguir ilustra a utilização de ST_Transform para converter uma geometria de um sistema de referência espacial em outro.

Primeiro, é criado o sistema de referência espacial plano de estado com um ID 3 utilizando uma chamada para db2se.

```
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
```

Em seguida, são incluídos pontos em:

- Tabela SAMPLE_POINTS_SP nas coordenadas planas de estado utilizando esse sistema de referência espacial.
- Tabela SAMPLE_POINTS_LL utilizando as coordenadas especificadas na latitude e longitude.

```
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) )
```

Em seguida, a função ST_Transform é utilizada para converter as geometrias.

Exemplo 2

Este exemplo converte pontos que estão nas coordenadas de latitude e longitude em coordenadas planas 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)
```

Exemplo 3

Este exemplo converte pontos que estão nas coordenadas planas de estado em coordenadas de latitude e longitude.

```
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)
```

Função ST_Union

A função ST_Union usa duas geometrias como parâmetros de entrada e retorna a geometria que é a união das geometrias especificadas. A geometria resultante é representada no sistema de referência espacial da primeira geometria.

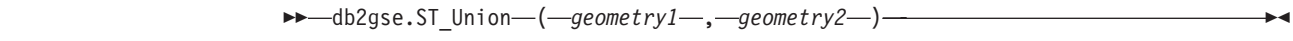
As duas geometrias devem ter a mesma dimensão. Se qualquer uma das duas geometrias especificadas for nula, será retornado nulo.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria e usar o mesmo datum subjacente, ela será convertida no outro sistema de referência espacial.

A geometria resultante é representada no tipo espacial mais apropriado. Se puder ser representada como um ponto, sequência de linhas ou polígono, será utilizado um desses tipos. Caso contrário, será utilizado o tipo multiponto, sequência multilinha ou multipolígono.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

geometry1
Um valor do tipo ST_Geometry ou um de seus subtipos que é combinado com *geometry2*.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que é combinado com *geometry1*.

Tipo de retorno

db2gse.ST_Geometry

Exemplos

Exemplo 1

As seguintes instruções SQL criam e ocupam a tabela 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))
```

Nos exemplos a seguir, os resultados foram reformatados para possibilitar a leitura. Seus resultados variarão, de acordo com sua exibição.

Exemplo 2

Este exemplo encontra a união de dois polígonos separados.

```
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)))
```

Exemplo 3

Este exemplo encontra a união de dois polígonos que fazem interseção.

```
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))
```

Exemplo 4

Localizar a união de duas sequências de linhas.

```
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
```

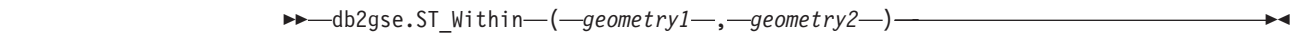
Resultados:

```
ID    ID    UNION
--- --- ------------------------------------------------------------
  4   5 MULTILINESTRING((70.00000000 70.00000000,80.00000000 80.00000000),
          (80.00000000 80.00000000,100.00000000 70.00000000))
```

Função ST_Within

Use a função ST_Within para determinar se uma geometria está totalmente dentro de outra geometria.

Sintaxe



Parâmetros

geometry1
Um valor do tipo ST_Geometry ou um de seus subtipos que deve ser testado para que esteja totalmente dentro de *geometry2*.

geometry2
Um valor do tipo ST_Geometry ou um de seus subtipos que deve ser testado para que esteja totalmente dentro de *geometry1*.

Tipo de retorno

INTEGER

Uso

ST_Within utiliza duas geometrias como parâmetros de entrada e retorna 1 se a primeira geometria estiver totalmente dentro da segunda. Caso contrário, será retornado 0 (zero).

Se qualquer uma das geometrias especificadas for nula ou vazia, será retornado nulo.

Se a segunda geometria não for representada no mesmo sistema de referência espacial que a primeira geometria e usar o mesmo datum subjacente, ela será convertida no outro sistema de referência espacial.

ST_Within executa a mesma operação lógica que ST_Contains executa com os parâmetros reversos. A função ST_Within retorna o resultado oposto exato da função ST_Contains.

A matriz de padrão da função ST_Within indica que os interiores das duas geometrias devem se cruzar e que o interior ou limite da geometria principal (geometria *a*) não deve cruzar o exterior da geometria secundária (geometria *b*). O asterisco (*) indica que as demais interseções não importam.

Tabela 32. Matriz para ST_Within

	Interior da Geometria b	**Limite da Geometria b**	**Exterior da Geometria b**
Interior da Geometria a	T	*	F
Limite da Geometria a	*	*	F
Exterior da Geometria a	*	*	*

Exemplos

A Figura 20 na página 409 mostra exemplos de ST_Within:

- Uma geometria de ponto está dentro de uma geometria multiponto quando seu interior cruza um dos pontos na segunda geometria.
- Uma geometria multiponto está dentro de uma geometria multiponto quando os interiores de todos os pontos cruzam a segunda geometria.
- Uma geometria multiponto está dentro de uma geometria de polígono quando todos os pontos estão no limite do polígono ou no interior do polígono.
- Uma geometria de ponto está dentro de uma geometria de sequência de linhas quando todos os pontos estão dentro da segunda geometria. Na Figura 20 na página 409, o ponto não está dentro da sequência de linhas porque seu interior não cruza a sequência de linhas; no entanto, a geometria multiponto está dentro da sequência de linhas porque todos os seus pontos cruzam o interior da sequência de linhas.
- Uma geometria de sequência de linhas está dentro de outras geometrias de sequência de linhas quando todos os seus pontos cruzam a segunda geometria.
- Uma geometria de ponto não está dentro de uma geometria de polígono porque seu interior não cruza o limite ou o interior do polígono.
- Uma geometria de sequência de linhas está dentro de uma geometria de polígono quando todos os seus pontos cruzam o limite ou o interior do polígono.
- Uma geometria de polígono está dentro de uma geometria de polígono quando todos os seus pontos cruzam o limite ou o interior do polígono.

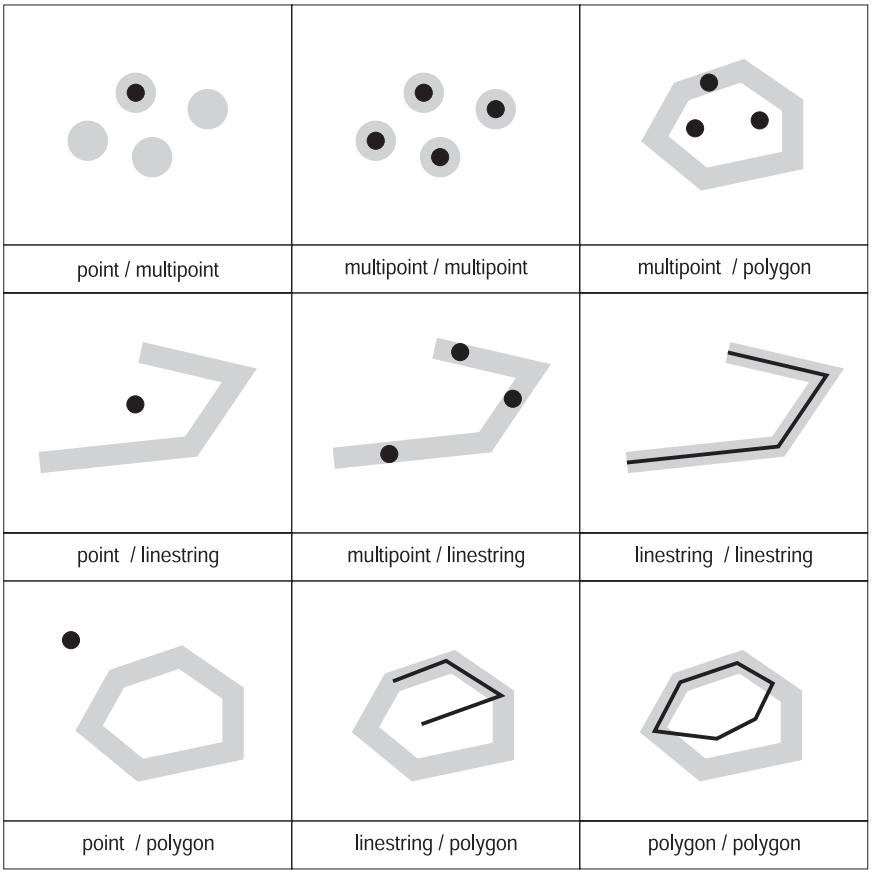


Figura 20. Função ST_Within

Exemplo 1

Este exemplo ilustra a utilização da função ST_Within. As geometrias são criadas e inseridas em três tabelas, SAMPLE_POINTS, SAMPLE_LINES e 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) )
```

Exemplo 2

Este exemplo encontra pontos na tabela SAMPLE_POINTS que estão nos polígonos na tabela 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
```

Exemplo 3

Este exemplo encontra sequências de linhas da tabela SAMPLE_LINES que estão nos polígonos da tabela 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
```

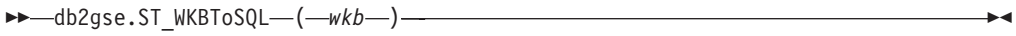
Função ST_WKBToSQL

A função ST_WKBToSQL usa uma representação binária reconhecida de uma geometria e retorna a geometria correspondente. O sistema de referência espacial com o identificador 0 (zero) é utilizado para a geometria resultante.

Se representação binária reconhecida especificada for nula, então nulo é retornado.

ST_WKBToSQL(*wkb*) fornece o mesmo resultado que ST_Geometry(*wkb*,0). A utilização da função ST_Geometry é recomendada sobre a utilização de ST_WKBToSQL por sua flexibilidade: ST_Geometry utiliza formatos adicionais de entrada, além da representação binária reconhecida.

Sintaxe



Parâmetro

wkb Um valor do tipo BLOB(2G) que contém a representação binária reconhecida da geometria resultante.

Tipo de retorno

db2gse.ST_Geometry

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra a utilização da função ST_WKBToSQL. Primeiro, as geometrias são armazenadas na tabela SAMPLE_GEOMETRIES em sua coluna GEOMETRY. Em seguida, suas representações binárias reconhecidas são armazenadas na coluna WKB utilizando a função ST_AsBinary na instrução UPDATE. Por último, a função ST_WKBToSQL é utilizada para retornar as coordenadas das geometrias na coluna 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
```

Utilize esta instrução SELECT para ver as geometrias na coluna 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))
```

Função ST_WKTToSQL

A função ST_WKTToSQL usa uma representação well-known text de uma geometria e retorna a geometria correspondente.

O sistema de referência espacial com o identificador 0 (zero) é utilizado para a geometria resultante.

Se a representação de texto reconhecida for nula, então nulo é retornado.

ST_WKTToSQL(*wkt*) fornece o mesmo resultado que ST_Geometry(*wkt*,0). O uso da função ST_Geometry oferece maior flexibilidade que a função ST_WKTToSQL porque ST_Geometry usa formas adicionais como entrada, bem como a representação well-known text.

Sintaxe

Parâmetro

wkt Um valor do tipo CLOB(2G) que contém a representação de texto reconhecida da geometria resultante.

Tipo de retorno

db2gse.ST_Geometry

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como ST_WKTToSQL pode criar e inserir geometrias utilizando suas representações de texto reconhecidas.

```
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 instrução SELECT retorna as geometrias que foram inseridas.

```
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))
```

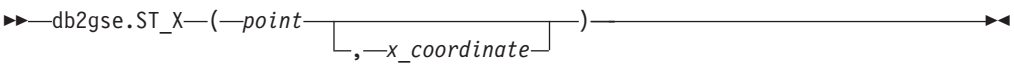
Função ST_X

A função ST_X toma um ponto como um parâmetro de entrada e retorna sua coordenada X. Como opção, é possível indicar uma coordenada X como parâmetro de entrada além do ponto e a função retorna o próprio ponto como sua coordenada X configurada como o valor determinado.

Se o ponto especificado for nulo ou vazio, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

ponto Um valor do tipo ST_Point para o qual a coordenada X é retornada ou modificada.

x_coordinate
Um valor do tipo DOUBLE que representa a nova coordenada X para *point*.

Tipos de retornos

- DOUBLE, se *x_coordinate* não for especificado
- db2gse.ST_Point, se *x_coordinate* for especificado

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_X. As geometrias são criadas e inseridas na tabela 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) )
```

Exemplo 2

Este exemplo encontra as coordenadas X dos pontos na tabela.

```
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
```

Exemplo 3

Este exemplo retorna um ponto com sua coordenada X definida como 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)
```

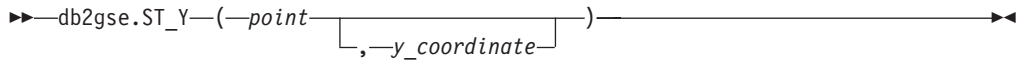
Função ST_Y

A função ST_Y toma um ponto como um parâmetro de entrada e retorna sua coordenada Y. Como opção, é possível indicar uma coordenada Y como parâmetro de entrada além do ponto e a função retorna o próprio ponto com sua coordenada Y configurada como o valor determinado.

Se o ponto especificado for nulo ou vazio, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

ponto Um valor do tipo ST_Point para o qual a coordenada Y é retornada ou modificada.

y_coordinate
Um valor do tipo DOUBLE que representa a nova coordenada Y para *point*.

Tipos de retornos

- DOUBLE, se *y_coordinate* não for especificado
- db2gse.ST_Point, se *y_coordinate* for especificado

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_Y. As geometrias são criadas e inseridas na tabela 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) )
```

Exemplo 2

Este exemplo encontra as coordenadas Y dos pontos na tabela.

```
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
```

Exemplo 3

Este exemplo retorna um ponto com sua coordenada Y definida como 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)
```

Função ST_Z

A função ST_Z toma um ponto como um parâmetro de entrada e retorna sua coordenada Y. Como opção, é possível indicar uma coordenada Z como parâmetro de entrada além do ponto e a função retorna o próprio ponto com sua coordenada Y configurada como o valor determinado.

ST_Z utiliza:

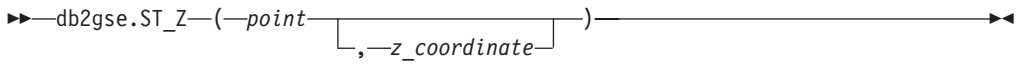
- Um ponto como parâmetro de entrada e retorna sua coordenada Z
- Um ponto e uma coordenada Z e retorna o próprio ponto com sua coordenada Z definida como o valor especificado, mesmo que o ponto especificado não tenha nenhuma coordenada Z existente.

Se a coordenada Z especificada for nula, a coordenada Z será removida do ponto.

Se o ponto especificado for nulo ou vazio, será retornado nulo.

Esta função também pode ser chamada como um método.

Sintaxe



Parâmetros

ponto Um valor do tipo ST_Point para o qual a coordenada Z é retornada ou modificada.

z_coordinate
Um valor do tipo DOUBLE que representa a nova coordenada Z para *point*.

Se *z_coordinate* for nulo, a coordenada Z será removida de *point*.

Tipos de retornos

- DOUBLE, se *z_coordinate* não for especificado
- db2gse.ST_Point, se *z_coordinate* for especificado

Exemplos

Exemplo 1

Este exemplo ilustra a utilização da função ST_Z. As geometrias são criadas e inseridas na tabela 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) )
```

Exemplo 2

Este exemplo encontra as coordenadas Z dos pontos na tabela.

```
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
```

Exemplo 3

Este exemplo retorna um ponto com sua coordenada Z definida como 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)
```

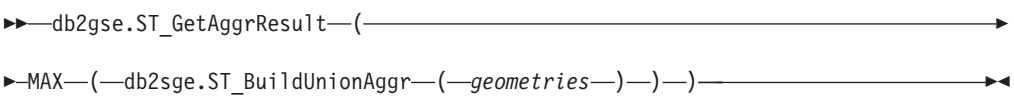
Funções Agregadas de União

Um agregado de união é a combinação das funções ST_BuildUnionAggr e ST_GetAggrResult. Use esta combinação para agregar uma coluna de geometrias em uma tabela a uma única geometria, construindo a união.

Se todas as geometrias a serem combinadas na união forem nulas, será retornado nulo. Se cada uma das geometrias a serem combinadas na união forem nulas ou vazias, será retornada uma geometria vazia do tipo ST_Point.

A função ST_BuildUnionAggr também pode ser chamada como um método.

Sintaxe



Parâmetros

MVS/ESA

Uma coluna em uma tabela do tipo ST_Geometry ou um dos seus subtipos e representa todas as geometrias que devem ser combinadas em uma união.

Tipo de retorno

db2gse.ST_Geometry

Restrições

Não é possível construir o agregado de união de uma coluna espacial em uma tabela em qualquer uma das seguintes situações:

- Em ambientes de banco de dados particionado
- Se uma cláusula GROUP BY for utilizada na seleção
- Se você utilizar uma função diferente da função agregada do DB2 MAX

Exemplo

No exemplo a seguir, as linhas de resultados foram formatadas novamente para possibilidade de leitura. O espaçamento nos resultados varia de acordo com a exibição on-line.

Este exemplo ilustra como um agregado de união pode ser utilizado para combinar um conjunto de pontos em multipontos. Diversos pontos são incluídos na tabela SAMPLE_POINTS. As funções ST_GetAggrResult e ST_BuildUnionAggr são utilizadas para construir a união dos pontos.

```
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)
```

Capítulo 19. Grupos de transformação

O Spatial Extender fornece quatro grupos de transformação que são utilizados para transferir geometrias entre o servidor DB2 e um aplicativo cliente.

Esses grupos de transformação acomodam os seguintes formatos de troca de dados:

- Representação WKT (well-known text)
- Representação WKB (well-known binary)
- Representação de ESRI shape
- GML (Geography Markup Language)

Quando os dados são recuperados de uma tabela que contém uma coluna espacial, os dados dessa coluna são transformados em um tipo de dados CLOB(2G) ou BLOB(2G), dependendo de você ter indicado se os dados transformados deviam ser representados em formato binário ou de texto. Também é possível utilizar os grupos de transformação para transferir dados espaciais no banco de dados.

Para selecionar qual grupo de transformação deve ser utilizado quando os dados são transferidos, utilize a instrução SET CURRENT DEFAULT TRANSFORM GROUP para modificar o registro especial CURRENT DEFAULT TRANSFORM GROUP do DB2. O DB2 utiliza o valor desse registro especial para determinar quais funções de transformação devem ser chamadas para executar as conversões necessárias.

Os grupos de transformação podem simplificar a programação dos aplicativos. Em vez de utilizar explicitamente as funções de conversão nas instruções SQL, você pode especificar um grupo de transformação, que permite que o DB2 cuide dessa tarefa.

Grupo de transformação ST_WellKnownText

Use o grupo de transformações ST_WellKnownText para transmitir dados para e a partir do DB2 usando a representação de WKT (texto reconhecido).

Ao ligar um valor do servidor de banco de dados ao cliente, a mesma função fornecida por ST_AsText() é utilizada para converter uma geometria na representação de WKT. Quando a representação de texto reconhecido de uma geometria é transferida para o servidor de banco de dados, a função ST_Geometry(CLOB) é utilizada implicitamente para executar as conversões em um valor ST_Geometry. Utilizar o grupo de transformação para ligar valores ao DB2 faz com que as geometrias sejam representadas no sistema de referência espacial com o identificador numérico 0 (zero).

Exemplos

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados pode variar de acordo com a exibição on-line.

Exemplo 1

O script SQL a seguir mostra como utilizar o grupo de transformação ST_WellKnownText para recuperar uma geometria em sua representação de texto reconhecido sem utilizar a função 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)
```

Exemplo 2

O código C a seguir mostra como utilizar o grupo de transformação ST_WellKnownText para inserir geometrias utilizando a função explícita ST_Geometry para a variável de host `wkt_buffer`, que tem o tipo CLOB e contém a representação de texto reconhecido do ponto (10 10) que deve ser inserido.

```
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 o grupo de transformação de todas as instruções SQL subsequentes
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);

// inserir o ponto utilizando WKT na coluna do tipo ST_Geometry
EXEC SQL
   INSERT
    INTO   transforms_sample(id, geom)
    VALUES (:id, :wkt_buffer);
```

Grupo de transformação ST_WellKnownBinary

Use o grupo de transformação ST_WellKnownBinary para transmitir dados de e para DB2 utilizando a representação de WKB (binário reconhecido).

Ao ligar um valor do servidor de banco de dados ao cliente, a mesma função fornecida por ST_AsBinary() é utilizada para converter uma geometria na representação de WKB. Quando a representação de binário reconhecido de uma geometria é transferida para o servidor de banco de dados, a função ST_Geometry(BLOB) é utilizada implicitamente para executar as conversões em um valor ST_Geometry. Utilizar o grupo de transformação para ligar valores ao DB2 faz com que as geometrias sejam representadas no sistema de referência espacial com o identificador numérico 0 (zero).

Exemplo

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados pode variar de acordo com a exibição on-line.

Exemplo 1

O script SQL a seguir mostra como utilizar o grupo de transformação ST_WellKnownBinary para recuperar uma geometria em sua representação de binário reconhecido sem utilizar a função 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
       0000000000244000000000000024400000000000003440
       0000000000003440000000000000344000000000000034
       4000000000000024400000000000002440000000000000
       2440'
```

Exemplo 2

O código C a seguir mostra como utilizar o grupo de transformação ST_WellKnownBinary para inserir geometrias utilizando a função explícita ST_Geometry para a variável de host `wkb_buffer`, que tem o tipo BLOB e contém a representação de binário reconhecido de uma geometria que deve ser inserida.

```
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 o grupo de transformação de todas as instruções SQL subsequentes
EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownBinary;

// inicializar variáveis do host
...
// inserir geometria utilizando WKB na coluna do tipo ST_Geometry

EXEC SQL
   INSERT
    INTO transforms_sample(id, geom)
    VALUES ( :id, :wkb_buffer );
```

Grupo de transformação ST_Shape

Use o grupo de transformação ST_Shape para transmitir dados de e para o DB2 utilizando a representação de formato ESRI.

Ao ligar um valor do servidor de banco de dados ao cliente, a mesma função fornecida por ST_AsShape() é utilizada para converter uma geometria em sua representação de forma. Ao transferir a representação de forma de uma geometria para o servidor de banco de dados, a função ST_Geometry(BLOB) é utilizada implicitamente para executar as conversões em um valor ST_Geometry. Utilizar o grupo de transformação para ligar valores ao DB2 faz com que as geometrias sejam representadas no sistema de referência espacial com o identificador numérico 0 (zero).

Exemplos

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados pode variar de acordo com a exibição on-line.

Exemplo 1

O script SQL a seguir mostra como o grupo de transformação ST_Shape pode ser utilizado para recuperar uma geometria em sua representação de forma sem utilizar a função 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'
```

Exemplo 2

O código C a seguir mostra como utilizar o grupo de transformação ST_Shape para inserir geometrias utilizando a função explícita ST_Geometry para a variável de host `shape_buffer`, que tem o tipo BLOB e contém a representação de forma de uma geometria que deve ser inserida.

```
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 o grupo de transformação de todas as instruções SQL subsequentes
EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// inicializar variáveis do host
...
```

```
SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// inserir geometria usando representação de forma
// na coluna de tipo ST_Geometry
EXEC SQL
     INSERT
    INTO transforms_sample(id, geom)
    VALUES ( :id, :shape_buffer );
```

Grupo de transformação ST_GML

Use o grupo de transformação ST_GML para transmitir dados de e para o DB2 utilizando GML (geography markup language).

Ao ligar um valor do servidor de banco de dados ao cliente, a mesma função fornecida por ST_AsGML() é utilizada para converter uma geometria em sua representação GML. Quando a representação GML de uma geometria é transferida para o servidor de banco de dados, a função ST_Geometry(CLOB) é utilizada implicitamente para executar as conversões em um valor ST_Geometry. Utilizar o grupo de transformação para ligar valores ao DB2 faz com que as geometrias sejam representadas no sistema de referência espacial com o identificador numérico 0 (zero).

Exemplos

Nos exemplos a seguir, as linhas de resultados foram reformatadas para facilitar a leitura. O espaçamento nos resultados pode variar de acordo com a exibição on-line.

Exemplo 1

O script SQL a seguir mostra como o grupo de transformação ST_GML pode ser utilizado para recuperar uma geometria em sua representação GML sem utilizar a função 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>
```

Exemplo 2

O código C a seguir mostra como utilizar o grupo de transformação ST_GML para inserir geometrias sem utilizar a função explícita ST_Geometry para a variável de host `gml_buffer`, que tem o tipo CLOB e contém a representação GML do ponto (20 ,20) que deve ser inserido.

```
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 o grupo de transformação de todas as instruções SQL subsequentes
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 variáveis do host
wkt_buffer.length = strlen(gml_buffer.data);

// inserir o ponto utilizando WKT na coluna do tipo ST_Geometry
EXEC SQL
   INSERT
   INTO transforms_sample(id, geom)
   VALUES ( :id, :gml_buffer );
```

Capítulo 20. Formatos de dados suportados

O DB2 Spatial Extender fornece formatos de dados espaciais padrão de mercado possíveis de usar.

São descritos os quatro formatos de dados espaciais a seguir:

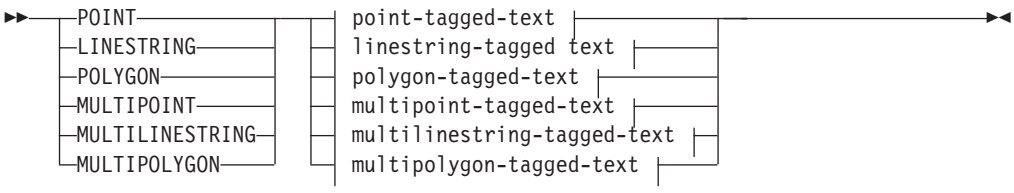
- Representação WKT (well-known text)
- Representação WKB (well-known binary)
- Representação de formatos
- Representação de GML (Geography Markup Language)

Representação WKT (well-known text)

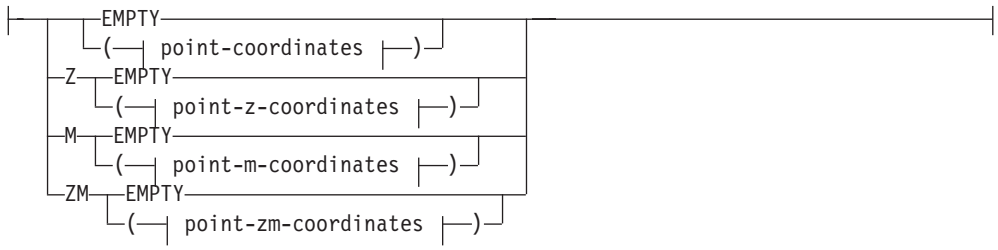
A especificação do OpenGIS Consortium "Simple Features for SQL" define a representação de texto reconhecida para trocar dados de geometrias em formato ASCII. Esta representação também é referida pelo padrão ISO "SQL/MM Part: 3 Spatial".

Consulte "Funções espaciais que convertem geometrias em e a partir de formatos de troca de dados" para obter informações sobre funções que aceitam e geram dados WKT.

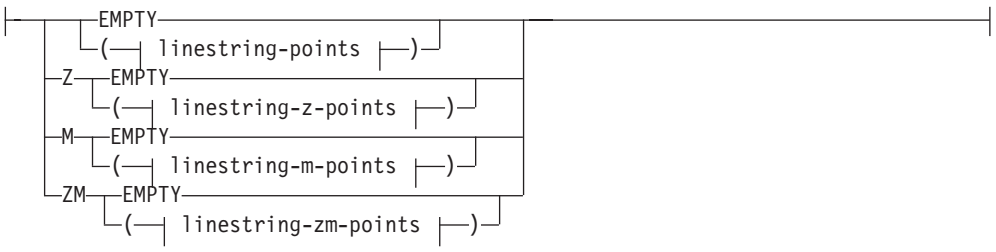
A representação de texto reconhecida de uma geometria é definida conforme a seguir:



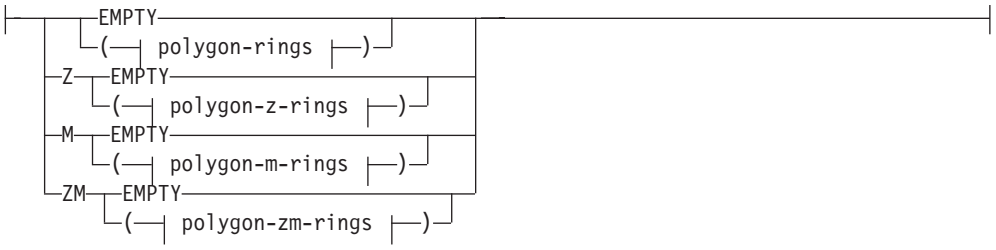
point-tagged-text:



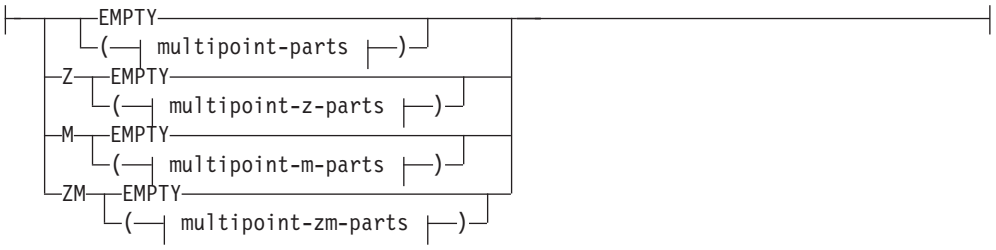
linestring-tagged-text:



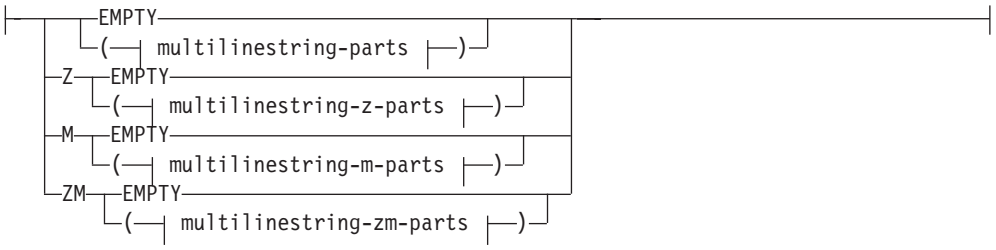
polygon-tagged-text:



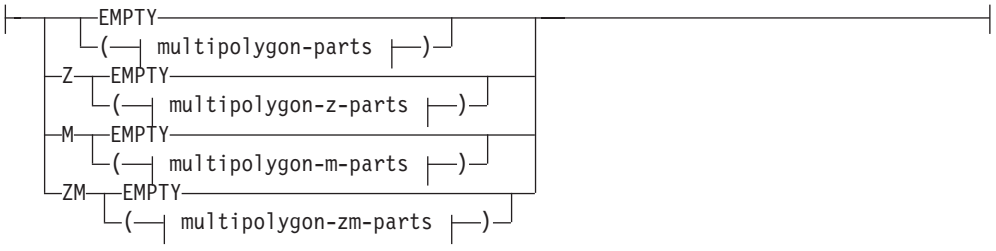
multipoint-tagged-text:



multilinestring-tagged-text:



multipolygon-tagged-text:

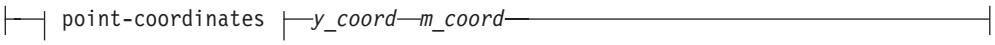


point-coordinates:

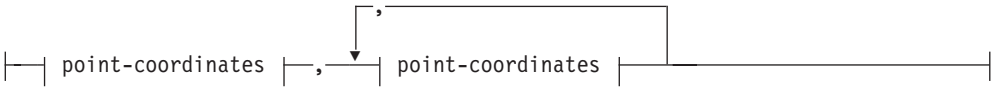
point-z-coordinates:

point-m-coordinates:

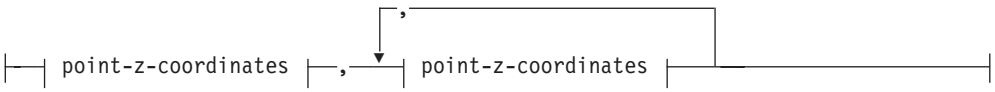
point-zm-coordinates:



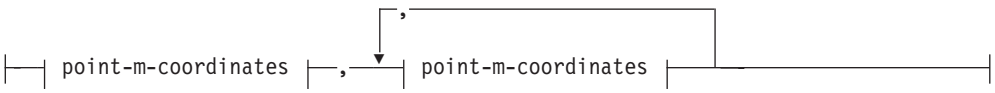
linestring-points:



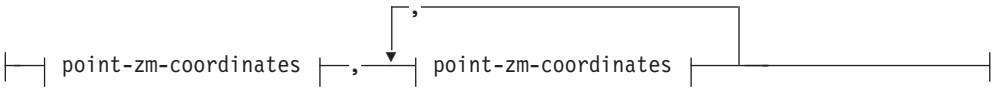
linestring-z-points:



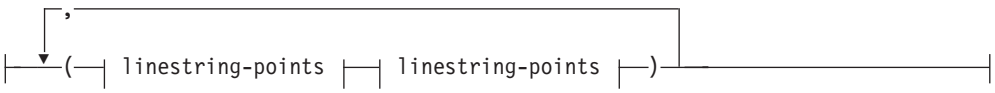
linestring-m-points:



linestring-zm-points:



polygon-rings:



polygon-z-rings:

polygon-m-rings:

polygon-zm-rings:

multipoint-parts:

multipoint-z-parts:

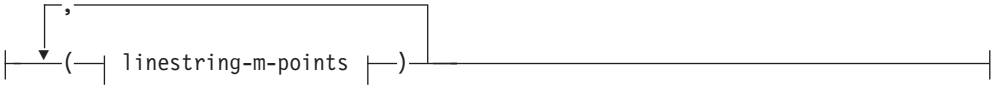
multipoint-m-parts:

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:

Parâmetros

x_coord

Um valor numérico (fixo, número inteiro ou vírgula flutuante), que representa a coordenada X de um ponto.

y_coord

Um valor numérico (fixo, número inteiro ou vírgula flutuante), que representa a coordenada Y de um ponto.

z_coord

Um valor numérico (fixo, número inteiro ou vírgula flutuante), que representa a coordenada Z de um ponto.

m_coord

Um valor numérico (fixo, número inteiro ou vírgula flutuante), que representa a coordenada M (medida) de um ponto.

Se a geometria estiver vazia, a palavra-chave EMPTY terá de ser especificada em vez da lista de coordenadas. A palavra-chave EMPTY não deve ser incorporada na lista de coordenadas

A tabela a seguir fornece alguns exemplos de possíveis representações de texto.

Tabela 33. Tipos de figura geométrica e suas representações de texto

Tipo de figura geométrica	Representação WKT	Comentário
ponto	POINT EMPTY	ponto vazio
ponto	POINT (10.05 10.28)	ponto
ponto	POINT Z(10.05 10.28 2.51)	ponto com coordenada Z
ponto	POINT M(10.05 10.28 4.72)	ponto com coordenada M
ponto	POINT ZM(10.05 10.28 2.51 4.72)	ponto com coordenada Z e coordenada M
sequência-de-linhas	LINESTRING EMPTY	sequência de linha vazia
polígono	POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))	polígono
multiponto	MULTIPOINT Z(10 10 2, 20 20 3)	multiponto com coordenadas Z
multilinestring	MULTILINESTRING M((310 30 1, 40 30 20, 50 20 10)(10 10 0, 20 20 1))	sequência multilinha com coordenadas M
multipolygon	MULTIPOLYGON ZM(((1 1 1 1, 1 2 3 4, 2 2 5 6, 2 1 7 8, 1 1 1 1)))	multipolígono com coordenadas Z e coordenadas M

Representação WKB (well-known binary)

Esta seção descreve a representação binária reconhecida para geometrias.

A especificação do OpenGIS Consortium "Simple Features for SQL" define a representação binária reconhecida. Esta representação também é definida pelo padrão International Organization for Standardization (ISO) "SQL/MM Part: 3 Spatial". Consulte a seção de referência relacionada no final deste tópico para obter informações sobre funções que aceitam e geram WKB.

O bloco de construção básica para representações binárias reconhecidas é o fluxo de bytes para um ponto, que consiste em dois valores duplos. Os fluxos de bytes

para outras figuras geométricas são construídos através de fluxos de bytes para figuras geométricas que já estejam definidas.

O exemplo a seguir ilustra o bloco de construção básica para representações binárias reconhecidas.

```
// 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
  Ponto    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;
  }
};
```

A figura a seguir mostra um exemplo de uma geometria na representação binária reconhecida utilizando a codificação NDR.

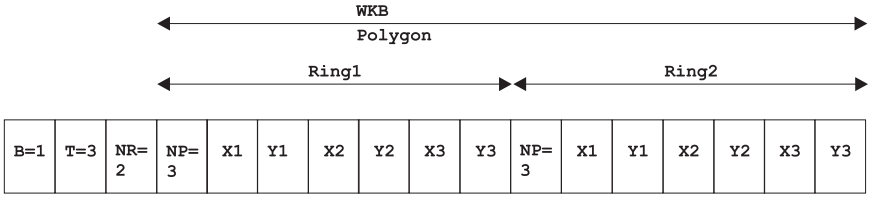


Figura 21. Representação geométrica em formato NDR. (B=1) do polígono tipo (T=3) com 2 lineares (NR=2), em que cada anel possui 3 pontos (NP=3).

Representação de formatos

A representação de formatos é um padrão da indústria amplamente utilizado, definido por ESRI.

Para obter uma descrição completa da representação de formatos, consulte o site do ESRI na Web no endereço http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

Representação de GML (Geography Markup Language)

A Linguagem de Marcação Geográfica (GML) é uma codificação XML para informações geográficas definidas pela especificação OpenGIS Consortium "Geography Markup Language V2".

O DB2 Spatial Extender possui várias funções que geram geometrias a partir de representações em representação GML (linguagem de marcação geográfica).

Para obter detalhes adicionais sobre esta especificação OpenGIS Consortium, consulte “OpenGIS Geography Markup Language (GML) Encoding Standard” em http://www.opengeospatial.org/standards/gml.

Capítulo 21. Sistemas de coordenadas suportados

O DB2 Spatial Extender usa uma sintaxe de sistemas de coordenadas específicos e valores do sistema de coordenadas suportados para fornecer uma representação textual padrão para informações do sistema de coordenadas.

Sintaxe de Sistemas de Coordenadas

A sintaxe de sistemas de coordenadas é uma representação em sequência desse sistema de coordenadas.

A representação de texto reconhecida dos sistemas de referência espacial fornece uma representação textual padrão para informações do sistema de coordenadas. As definições da representação de texto bem conhecido são definidas pela especificação OGC "Simple Features for SQL" e pelo ISO SQL/MM Part 3: Spatial standard.

Um sistema de coordenadas é um sistema de coordenadas geográficas (latitude-longitude), projetadas (X,Y) ou geocêntricas (X,Y,Z). O sistema de coordenadas é composto de vários objetos. Cada objeto tem uma palavra-chave em maiúsculas (por exemplo, DATUM ou UNIT) seguido dos parâmetros de definição, delimitados por vírgulas, do objeto entre colchetes. Alguns objetos são compostos de outros objetos, portanto, o resultado é uma estrutura aninhada.

Nota: As implementações estão livres para substituir os parênteses padrão ( ) por colchetes [ ] e devem poder ler os dois formatos de sinais gráficos.

A definição EBNF (Extended Backus Naur Form) para a representação em sequência de um sistema de coordenadas usando colchetes retos é a seguinte (consulte a nota anterior referente ao uso de colchetes):

```
<coordinate system> = <projected cs> |
<geographic cs> | <geocentric cs>
<projected cs> = PROJCS["<name>",
<geographic cs>, <projection>, {<parameter>,}*
<linear unit>]
<projection> = PROJECTION["<name>"]
<parameter> = PARAMETER["<name>",
<value>]

<value> = <number>
```

O tipo de sistema de coordenadas é identificado pela palavra-chave utilizada:

PROJCS
Um sistema de coordenadas do conjunto de dados será identificado pela palavra-chave PROJCS se os dados estiverem nas coordenadas projetadas

GEOGCS
Um sistema de coordenadas do conjunto de dados será identificado pela palavra-chave GEOGCS se os dados estiverem nas coordenadas geográficas

GEOCCS
Um sistema de coordenadas do conjunto de dados será identificado pela palavra-chave GEOGCS se os dados estiverem nas coordenadas geocêntricas

A palavra-chave PROJCS é seguida por todas as "partes" que definem o sistema de coordenadas projetadas. A primeira parte de qualquer objeto é sempre o nome. Vários objetos seguem o nome do sistema de coordenadas projetadas: o sistema de coordenadas geográficas, a projeção de mapas, um ou mais parâmetros e a unidade linear de medida. Todos os sistemas de coordenadas projetadas são baseados num sistema de coordenadas geográficas, portanto, esta sessão descreve primeiro as partes específicas de um sistema de coordenadas projetadas. Por exemplo, a zona UTM 10N nos dados NAD83 está definida:

```
PROJCS["NAD_1983_UTM_Zone_10N",
<geographic cs>,
PROJECTION["Transverse_Mercator"],
PARAMETER["False_Easting",500000.0],
PARAMETER["False_Northing",0.0],
PARAMETER["Central_Meridian",-123.0],
PARAMETER["Scale_Factor",0.9996],
PARAMETER["Latitude_of_Origin",0.0],
UNIT["Meter",1.0]]
```

O nome e vários objetos definem o objeto do sistema de coordenadas geográficas em turnos: o dado, o meridiano principal e a unidade angular de medida.

```
<geographic cs> = GEOGCS["<name>", <datum>, <prime meridian>, <angular unit>]
<datum> = DATUM["<name>", <spheroid>]
<spheroid> = SPHEROID["<name>", <semi-major axis>, <inverse flattening>]
<semi-major axis> = <number>
<inverse flattening> = <number>
<prime meridian> = PRIMEM["<name>", <longitude>]
<longitude> = <number>
```

O semi-eixo maior é medido em metros e deve ser maior do que zero.

A sequência do sistema de coordenadas geográficas para a zona UTM 10 em NAD83:

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],
UNIT["Degree",0.0174532925199433]]
```

O objeto UNIT pode representar unidade angular ou linear de medidas:

```
<angular unit> = <unit>
<linear unit> = <unit>
<unit> = UNIT["<name>", <conversion factor>]
<conversion factor> = <number>
```

O fator de conversão especifica o número de metros (para uma unidade linear) ou o número de radianos (para uma unidade angular) por unidade e deve ser maior que zero.

Assim, a representação completa da sequência da Zona UTM 10N é a seguinte:

```
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]]
```

Um sistema de coordenadas geométricas é semelhante a um sistema de coordenadas geográficas:

```
<geocentric cs> = GEOCCS["<name>", <datum>, <prime meridian>, <linear unit>]
```

Unidades lineares suportadas

Use unidades lineares que são suportadas pelo DB2 Spatial Extender.

Tabela 34. Unidades lineares suportadas

Unidade	Fator de conversão
Metro	1,0
Pé (Internacional)	0,3048
Pé americano	12/39,37
Pé americano modificado	12,0004584/39,37
Pé de Clarke	12/39,370432
Pé indiano	12/39,370141
Ligação	7,92/39,370432
Link (Benoit)	7,92/39,370113
Link (Sears)	7,92/39,370147
Cadeia (Benoit)	792/39,370113
Cadeia (Sears)	792/39,370147
Jarda (Índia)	36/39,370141
Jarda (Sears)	36/39,370147
Braça	1.8288
Milha náutica	1852.0

Unidades angulares suportadas

Use unidades angulares que são suportadas pelo DB2 Spatial Extender.

Tabela 35. Unidades angulares suportadas

Unidade	Intervalo válido para latitude	Intervalo válido para longitude	Fator de conversão
Radiano	radianos -pi/2 e pi/2 (inclusive)	radianos -pi e pi (inclusive)	1,0
Grau Decimal	-90 e 90 graus (inclusive)	-180 e 180 graus (inclusive)	pi/180
Minuto Decimal	-5400 e 5400 minutos (inclusive)	-10800 e 10800 minutos (inclusive)	(pi/180)/60
Segundo Decimal	-324000 e 324000 segundos (inclusive)	-648000 e 648000 segundos (inclusive)	(pi/180)*3600

Tabela 35. Unidades angulares suportadas (continuação)

Unidade	Intervalo válido para latitude	Intervalo válido para longitude	Fator de conversão
Gon	-100 e 100 gradianos (inclusive)	-200 e 200 gradianos (inclusive)	pi/200
Grade	-100 e 100 gradianos (inclusive)	-200 e 200 gradianos (inclusive)	pi/200

Esferóides suportados

Use esferóides que são suportados pelo DB2 Spatial Extender.

Tabela 36. Esferóides suportados

Nome	Eixo semi-principal	Condensação inversa
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

Tabela 36. Esferóides suportados (continuação)

Nome	Eixo semi-principal	Condensação inversa
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
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
Internacional 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

Meridianos principais suportados

Use meridianos principais que são suportados pelo DB2 Spatial Extender.

Tabela 37. Meridianos principais suportados

Local	Coordenadas
Greenwich	0° 0' 0"
Bern	7° 26' 22.5" E
Bogotá	74° 4' 51.3" W
Bruxelas	4° 22' 4.71" E
Ferro	17° 40' 0" W
Jakarta	106° 48' 27.79" E
Lisbon	9° 7' 54.862" W
Madri	3° 41' 16.58" W
Paris	2° 20' 14.025" E
Roma	12° 27' 8.4" E
Estocolmo	18° 3' 29" E

Projeções do mapa suportadas

Use projeções de mapa que são suportadas pelo DB2 Spatial Extender.

Tabela 38. Projeções cilíndricas

Projeções cilíndricas	Projeções pseudocilíndricas
Behrmann	Craster parabolic
Cassini	Eckert I
Área igual cilíndrica	Eckert II
Equiretangular	Eckert III
Estereográfico de Gall	Eckert IV
Gauss-Kruger	Eckert V
Mercator	Eckert VI
Miller cilindrico	McBryde-Thomas flat polar quartic
Oblíquo	Mercator (Hotine) Mollweide
Plate-Carrée	Robinson
Times	Sinusoidal (Sansom-Flamsteed)
Transverse Mercator	Winkel I

Tabela 39. Projeções cônicas

Nome	Projeção cônica
Área igual cônica de Albers	Chamberlin trimétrico
Cônico conformal oblíquo bipolar	Dois pontos equidistantes
Bonne	Área igual Hammer-Aitoff
Cônico equidistante	Van der Grinten I
Cônico conformal de Lambert	Diversos
Policônico	Alaska série E
Cônico simples	Grade Alaska (Estereográfico Modificado por Snyder)

Tabela 40. Parâmetros de projeção do mapa

Parâmetro	Descrição
central_meridian	A linha da longitude escolhida como a origem das coordenadas x.
scale_factor	Scale_factor geralmente é utilizado para reduzir a quantidade de distorção em uma projeção de mapa.
standard_parallel_1	Uma linha de latitude que geralmente não apresenta distorção. Usada também para "latitude de escala verdadeira."
standard_parallel_2	Uma linha de longitude que geralmente não apresenta distorção.
longitude_of_center	A longitude que define o ponto central da projeção do mapa.
latitude_of_center	A latitude que define o ponto central da projeção do mapa.
longitude_of_origin	A longitude escolhida como a origem das coordenadas x.
latitude_of_origin	A latitude escolhida como a origem das coordenadas y.
false_easting	Um valor incluído em coordenadas x para que todos os valores de coordenadas x sejam positivos.
false_northing	Um valor incluído em coordenadas y para que todas as coordenadas y sejam positivas.
azimute	O ângulo leste do norte que define a linha central de uma projeção oblíqua.
longitude_of_point_1	A longitude do primeiro ponto necessário para uma projeção de mapa.

Tabela 40. Parâmetros de projeção do mapa (continuação)

Parâmetro	Descrição
latitude_of_point_1	A latitude do primeiro ponto necessário para uma projeção de mapa.
longitude_of_point_2	A longitude do segundo ponto necessário para uma projeção de mapa.
latitude_of_point_2	A latitude do segundo ponto necessário para uma projeção de mapa.
longitude_of_point_3	A longitude do terceiro ponto necessário para uma projeção de mapa.
latitude_of_point_3	A latitude do terceiro ponto necessário para uma projeção de mapa.
landsat_number	O número de um satélite Landsat.
path_number	O número de caminho orbital de um determinado satélite.
perspective_point_height	O peso acima da terra do ponto de perspectiva da projeção do mapa.
fipszone	Número de zona do Sistema de Coordenadas Planas do Estado.
zona	Número de zona UTM.

Apêndice A. Visão Geral das Informações Técnicas do DB2

As informações técnicas do DB2 estão disponíveis em vários formatos que podem ser acessados de várias maneiras.

As informações técnicas do DB2 estão disponíveis por meio das ferramentas e métodos a seguir:

- DB2Centro de Informações
  - Tópicos (Tópicos de tarefa, conceito e referência)
  - Programas de amostra
  - Tutoriais
- Manuais do DB2
  - Arquivos PDF (por download)
  - Arquivos PDF (do DVD em PDF do DB2)
  - Manuais impressos
- Ajuda da linha de comandos
  - Ajuda do comando
  - Ajuda da mensagem

Nota: Os tópicos do Centro de Informações do DB2 são atualizados com mais frequência que o os manuais em PDF ou em cópia impressa. Para obter as informações mais atuais, instale as atualizações da documentação assim que elas forem disponibilizadas ou consulte o Centro de Informações do DB2 em ibm.com.

É possível acessar informações técnicas adicionais do DB2, como technotes, White Papers e publicações do IBM Redbooks on-line em ibm.com. Acesse o site de biblioteca de software do DB2 Information Management em http://www.ibm.com/software/data/sw-library/.

Feedback da Documentação

Nós apreciamos seu feedback sobre a documentação do DB2. Se você tiver sugestões sobre como melhorar a documentação do DB2, envie um e-mail para db2docs@ca.ibm.com. A equipe de documentação do DB2 lê todos os feedbacks, mas não pode responder diretamente para você. Forneça exemplos específicos sempre que possível, para que melhor possamos compreender suas preocupações. Se estiver enviando feedback sobre um tópico ou arquivo de ajuda específico, inclua o título do tópico e a URL.

Não use este endereço de e-mail para entrar em contato com o Suporte ao Cliente do DB2. Se você tiver um problema técnico com o DB2 que a documentação não resolva, entre em contato com o centro de atendimento IBM local para obter assistência.

Biblioteca Técnica do DB2 em Cópia Impressa ou em Formato PDF

As seguintes tabelas descrevem a biblioteca do DB2 disponível no IBM Publications Center em www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss. Os manuais do DB2 Versão 10.1 em inglês e traduzidos no formato PDF podem ser transferidos por download a partir de www.ibm.com/support/docview.wss?rs=71&uid=swg2700947.

Embora as tabelas identifiquem os manuais disponíveis em cópia impressa, é possível que não estejam disponíveis em seu país.

O número do formulário aumenta cada vez que um manual é atualizado. Certifique-se de que você esteja lendo a versão mais recente dos manuais, conforme listado abaixo.

Nota: O Centro de Informações do *DB2* é atualizado com mais frequência do que os manuais em PDF ou em cópia impressa.

Tabela 41. Informações Técnicas do DB2

Nome	Número do Formulário	Disponível em Cópia Impressa	Última atualização
Administrative API Reference	SC27-3864-00	Sim	Abril, 2012
Administrative Routines and Views	SC27-3865-00	Não	Abril, 2012
Guia e Referência da Interface do Nível de Chamada Volume 1	SC27-3866-00	Sim	Abril, 2012
Guia e Referência da Interface do Nível de Chamada Volume 2	SC27-3867-00	Sim	Abril, 2012
Command Reference	SC27-3868-00	Sim	Abril, 2012
Database Administration Concepts and Configuration Reference	SC27-3871-00	Sim	Abril, 2012
Data Movement Utilities Guide and Reference	SC27-3869-00	Sim	Abril, 2012
Database Monitoring Guide and Reference	SC27-3887-00	Sim	Abril, 2012
Data Recovery and High Availability Guide and Reference	SC27-3870-00	Sim	Abril, 2012
Database Security Guide	SC27-3872-00	Sim	Abril, 2012
DB2 Workload Management Guide and Reference	SC27-3891-00	Sim	Abril, 2012
Developing ADO.NET and OLE DB Applications	SC27-3873-00	Sim	Abril, 2012
Developing Embedded SQL Applications	SC27-3874-00	Sim	Abril, 2012

Tabela 41. Informações Técnicas do DB2 (continuação)

Nome	Número do Formulário	Disponível em Cópia Impressa	Última atualização
Developing Java Applications	SC27-3875-00	Sim	Abril, 2012
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-3876-00	Não	Abril, 2012
Developing User-defined Routines (SQL and External)	SC27-3877-00	Sim	Abril, 2012
Getting Started with Database Application Development	GI13-2046-00	Sim	Abril, 2012
Introdução à Instalação e Administração do DB2 no Linux e Windows	G517-0196-00	Sim	Abril, 2012
Globalization Guide	SC27-3878-00	Sim	Abril, 2012
Instalando Servidores DB2	G517-0195-00	Sim	Abril, 2012
Instalalando o IBM Data Server Clients	G517-0197-00	Não	Abril, 2012
Referência de Mensagens Volume 1	SC27-3879-00	Não	Abril, 2012
Referência de Mensagens Volume 2	SC27-3880-00	Não	Abril, 2012
Net Search Extender Administration and User's Guide	SC27-3895-00	Não	Abril, 2012
Partitioning and Clustering Guide	SC27-3882-00	Sim	Abril, 2012
pureXML Guide	SC27-3892-00	Sim	Abril, 2012
Referência e Guia do Usuário do Spatial Extender	S517-0064-00	Não	Abril, 2012
SQL Procedural Languages: Ativação e Suporte de Aplicativo	SC27-3896-00	Sim	Abril, 2012
SQL Reference Volume 1	SC27-3885-00	Sim	Abril, 2012
SQL Reference Volume 2	SC27-3886-00	Sim	Abril, 2012
Text Search Guide	SC27-3888-00	Sim	Abril, 2012
Troubleshooting and Tuning Database Performance	SC27-3889-00	Sim	Abril, 2012
Atualizando para DB2 Versão 10.1	S517-0017-00	Sim	Abril, 2012
O que Há de Novo para o DB2 Versão 10.1	S517-0063-00	Sim	Abril, 2012
XQuery Reference	SC27-3893-00	Não	Abril, 2012

Tabela 42. Informações Técnicas Específicas do DB2 Connect

Nome	Número do Formulário	Disponível em Cópia Impressa	Última atualização
DB2 Connect Instalando e Configurando o DB2 Connect Personal Edition	SC27-3861-00	Sim	Abril, 2012
DB2 Connect Instalando e Configurando Servidores DB2 Connect	SC27-3862-00	Sim	Abril, 2012
DB2 Connect User's Guide	SC27-3863-00	Sim	Abril, 2012

Exibindo Ajuda de Estado SQL a partir do Processador de Linha de Comando

Os produtos do DB2 retornam um valor SQLSTATE para condições que podem ser o resultado de uma instrução SQL. A ajuda de SQLSTATE explica os significados de estados de SQL e de códigos de classe de estado de SQL.

Procedimento

Para iniciar a ajuda de estado de SQL, abra o processador da linha de comandos e insira:

```
? sqlstate ou ? class code
```

, em que *sqlstate* representa um estado SQL válido de cinco dígitos e *class code* representa os primeiros dois dígitos do estado SQL.
Por exemplo, `? 08003` exibe a ajuda para o estado de SQL 08003 e `? 08` exibe o auxílio para o código de classe 08.

Acessando Diferentes Versões do Centro de Informações do DB2

A documentação para outras versões de produtos do DB2 é localizada em centros de informações separados em ibm.com.

Sobre Esta Tarefa

Para tópicos do DB2 Versão 10.1, a URL do *Centro de Informações do DB2* é http://publib.boulder.ibm.com/infocenter/db2luw/v10r1.

Para tópicos do DB2 Versão 9.8, a URL do *Centro de Informações do DB2* é http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/.

Para tópicos do DB2 Versão 9.7, a URL do *Centro de Informações do DB2* é http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/.

Para tópicos do DB2 Versão 9.5, a URL do *Centro de Informações do DB2* é http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Para tópicos do DB2 Versão 9.1, a URL do *Centro de Informações do DB2* é http://publib.boulder.ibm.com/infocenter/db2luw/v9/.

Para tópicos do DB2 Versão 8, acesse a URL do *Centro de Informações do DB2* em: http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

Atualizando o Centro de Informações do DB2 Instalado no seu Computador ou Servidor de Intranet

Um Centro de Informações do DB2 instalado localmente deve ser atualizado periodicamente.

Antes de Iniciar

Um Centro de Informações do DB2 Versão 10.1 já deve estar instalado. Para obter detalhes, consulte "Instalando o Centro de Informações do DB2 usando o tópico Assistente de Configuração do DB2" em *Instalando Servidores DB2*. Todos os pré-requisitos e restrições que se aplicam à instalação do Centro de Informações também se aplicam à atualização do Cento de Informações.

Sobre Esta Tarefa

Um Centro de Informações do DB2 existente pode ser atualizado automática ou manualmente:

- As atualizações automáticas atualizam recursos e idiomas existentes do Centro de Informações. Um benefício das atualizações automáticas é que o Centro de Informações está indisponível por um tempo mais curto em comparação com durante uma atualização manual. Além disso, as atualizações automáticas podem ser configuradas para executar como parte de outras tarefas em lote que executam periodicamente.
- As atualizações manuais podem ser usadas para atualizar recursos e idiomas existentes do Centro de Informações. As atualizações automáticas reduzem o tempo de inatividade durante o processo de atualização, porém, você deve usar o processo manual quando desejar incluir recursos ou idiomas. Por exemplo, um Centro de Informações local foi originalmente instalado com ambos os idiomas, inglês e francês, e agora você também deseja instalar o idioma alemão; uma atualização manual instalará o alemão, assim como atualizará os recursos e idiomas do Centro de Informações existente. Porém, uma atualização manual necessita que o Centro de Informações seja manualmente parado, atualizado e reiniciado. O Centro de Informações permanece indisponível durante o processo de atualização inteiro. No processo de atualização automática, o Centro de Informações fica indisponível para reiniciar o Centro de Informações apenas depois da atualização.

Este tópico detalha o processo para atualizações automáticas. Para instruções de atualizações manuais, consulte o tópico "Instalando manualmente o Centro de Informações doDB2 instalado no seu computador ou servidor de intranet".

Procedimento

Para atualizar automaticamente o Centro de Informações do DB2 instalado em seu computador ou servidor de intranet:

1. Em sistemas operacionais Linux,
   a. Navegue até o caminho onde o Centro de Informações está instalado. Por padrão, o Centro de Informações do DB2 é instalado no diretório `/opt/ibm/db2ic/V10.1`.
   b. Navegue do diretório de instalação para o diretório `doc/bin`.

c. Execute o script `update-ic`:

   ```
   update-ic
   ```

2. Em sistemas operacionais Windows,
   a. Abra uma janela de comandos.
   b. Navegue até o caminho onde o Centro de Informações está instalado. Por padrão, o Centro de Informações do DB2 é instalado no diretório `<Program Files>\IBM\Centro de Informações do DB2\Versão 10.1`,em que `<Program Files>` representa o local do diretório Program Files.
   c. Navegue do diretório de instalação para o diretório `doc\bin`.
   d. Execute o arquivo `update-ic.bat`:

      ```
      update-ic.bat
      ```

Resultados

O Centro de Informações do DB2 é reiniciado automaticamente. Se as atualizações estão disponíveis, o Centro de Informações exibe os tópicos novos e atualizados. Se as atualizações do Centro de Informações não estão disponíveis, uma mensagem é adicionado ao log. O arquivo de log está localizado no diretório `doc\eclipse\configuration`. O nome do arquivo de log é um número gerado aleatoriamente. Por exemplo, `1239053440785.log`.

Atualizando Manualmente o Centro de Informações do DB2 Instalado em seu Computador ou Servidor de Intranet

Se você instalou o Centro de Informações do DB2 localmente, é possível obter e instalar atualizações de documentações da IBM.

Sobre Esta Tarefa

Atualizar manualmente o *Centro de Informações do DB2* instalado localmente requer que você:

1. Pare o *Centro de Informações do DB2* em seu computador e reinicie o Centro de Informações em modo independente. Executar o Centro de Informações no modo independente impede que outros usuários em sua rede o acessem, e permite que você aplique atualizações. O Versão Workstation do Centro de Informações do DB2 sempre é executado no modo independente. .
2. Utilize o recurso de Atualização para verificar quais atualizações estão disponíveis. Se houver atualizações que você deve instalar, é possível utilizar o recurso Atualizar para obter e instalá-las

   Nota: Se seu ambiente precisar da instalação de atualizações do *Centro de Informações do DB2* em uma máquina que não esteja conectada à Internet, espelhe o site de atualização em um sistema de arquivos local usando uma máquina que esteja conectada à Internet e que tenha o Centro de Informações do *DB2* instalado. Se muitos usuários em sua rede estiverem instalando as atualizações da documentação, será possível reduzir o tempo necessário para que os indivíduos façam as atualizações, espelhando também o site de atualização localmente e criando um proxy para o site de atualização.
   Se houver pacotes de atualização disponíveis, utilize o recurso Update para obter os pacotes. No entanto, o recursos Atualização está disponível apenas no modo independente.
3. Pare o Centro de Informações independente e reinicie o *Centro de Informações do DB2* em seu computador.

Nota: No Windows 2008, Windows Vista (e superior), os comandos listados posteriormente nesta seção deverão ser executados como um administrador. Para abrir um prompt de comandos ou ferramenta gráfica com privilégios totais de administrador, clique com o botão direito no atalho e, em seguida, selecione **Executar como Administrador**.

Procedimento

Para atualizar o *Centro de Informações do DB2* instalado em seu computador ou servidor de intranet:

1. Pare o *Centro de Informações do DB2*.
   - No Windows, clique em **Iniciar** > **Painel de Controle** > **Ferramentas Administrativas** > **Serviços**. Em seguida, clique com o botão direito no serviço **Centro de Informações do DB2** e selecione **Parar**.
   - No Linux, digite o seguinte comando:
     ```
     /etc/init.d/db2icdv10 stop
     ```
2. Inicie o Centro de Informações no modo independente.
   - No Windows:
     a. Abra uma janela de comandos.
     b. Navegue até o caminho onde o Centro de Informações está instalado. Por padrão, o Centro de Informações do *DB2* é instalado no diretório *Program_Files*\IBM\Centro de Informações do DB2\Versão 10.1, em que *Program_Files* representa o local do diretório Arquivos de Programas.
     c. Navegue do diretório de instalação para o diretório doc\bin.
     d. Execute o arquivo help_start.bat:
        ```
        help_start.bat
        ```
   - No Linux:
     a. Navegue até o caminho onde o Centro de Informações está instalado. Por padrão, o *Centro de Informações do DB2* está instalado no diretório /opt/ibm/db2ic/V10.1.
     b. Navegue do diretório de instalação para o diretório doc/bin.
     c. Execute o script help_start:
        ```
        help_start
        ```

   O navegador da Web padrão dos sistemas é aberto para exibir o Centro de Informações independente.
3. Clique no botão **Atualizar** (). (JavaScript deve estar ativado em seu navegador.) No painel direito do Centro de Informações, clique em **Localizar Atualizações**. Será exibida uma lista com atualizações para a documentação existente.
4. Para iniciar o processo de instalação, verifique as seleções que deseja instalar e, em seguida, clique em **Instalar Atualizações**.
5. Após a conclusão do processo de instalação, clique em **Concluir**.
6. Pare o Centro de Informações independente:
   - No Windows, navegue até o diretório doc\bin dentro do diretório de instalação e execute o arquivo help_end.bat:
     ```
     help_end.bat
     ```

Nota: O arquivo em lote `help_end` contém os comandos necessários para parar com segurança os processos que foram iniciados com o arquivo em lote `help_start`. Não utilize `Ctrl-C` ou qualquer outro método para parar `help_start.bat`.

- No Linux, navegue até o diretório `doc/bin` dentro do diretório de instalação e execute o script `help_end`:

  ```
  help_end
  ```

  Nota: O script `help_end` contém os comandos necessários para parar com segurança os processos que foram iniciados com o script `help_start`. Não utilize qualquer outro método para parar o script `help_start`.

7. Reinicie o *Centro de Informações do DB2*.
   - No Windows, clique em **Iniciar** > **Painel de Controle** > **Ferramentas Administrativas** > **Serviços**. Em seguida, clique com o botão direito no serviço **Centro de Informações do DB2** e selecione **Iniciar**.
   - No Linux, digite o seguinte comando:

     ```
     /etc/init.d/db2icdv10 start
     ```

Resultados

O *Centro de Informações do DB2* atualizado exibe os tópicos novos e atualizados.

Tutoriais do DB2

Os tutoriais do DB2 ajudam a aprender sobre vários aspectos dos produtos do banco de dados DB2. As lições oferecem instruções passo a passo.

Antes de iniciar

É possível visualizar a versão XHTML do tutorial do Centro de Informações em http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/.

Algumas lições utilizam dados ou código de amostra. Consulte o tutorial para obter uma descrição dos pré-requisitos para suas tarefas específicas.

Tutoriais do DB2

Para visualizar o tutorial, clique no título.

"pureXML" em *pureXML Guide*
: Configure um banco de dados DB2 para armazenar dados XML e executar operações básicas com o armazenamento de dados do XML nativo.

Informações sobre Resolução de Problemas do DB2

Uma grande variedade de informações sobre determinação e resolução de problemas está disponível para ajudá-lo a usar produtos de banco de dados DB2.

Documentação do DB2
: As informações sobre resolução de problemas podem ser localizadas no *Troubleshooting and Tuning Database Performance* ou na seção Fundamentos do Banco de Dados do Centro de Informações do *DB2* , que contém:
  - Informações sobre como isolar e identificar problemas com ferramentas e utilitários de diagnóstico do DB2.
  - Soluções para alguns dos problemas mais comuns.

- Conselho para ajudar a resolver outros problemas que podem ser encontrados com seus produtos de banco de dados DB2.

Portal de Suporte IBM

Consulte o Portal de Suporte IBM se estiver tendo problemas e quiser ajuda para localizar as possíveis causas e soluções. O site Suporte Técnico possui links para as publicações mais recentes do DB2, TechNotes, APARs (Authorized Program Analysis Reports) ou correções de erros, fix packs e outros recursos. Você pode pesquisar essa base de conhecimento para localizar as possíveis soluções para seus problemas.

Acesse o Portal de Suporte IBM em http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows

Termos e Condições

As permissões para uso destas publicações são concedidas sujeitas aos seguintes termos e condições.

Uso Pessoal: Será possível reproduzir estas Publicações apenas para uso pessoal e não comercial, contanto que todos os avisos do proprietário sejam preservados. O Cliente não deve distribuir, exibir ou criar trabalhos derivativos destas Publicações ou de qualquer parte delas, sem o consentimento expresso da IBM.

Uso Comercial O Cliente poderá reproduzir, distribuir e exibir essas Publicações somente dentro da empresa do Cliente, contanto que todos os avisos do proprietário sejam preservados. O Cliente não poderá criar trabalhos derivativos destas Publicações ou reproduzir, distribuir ou exibir estas Publicações ou qualquer parte delas fora de sua empresa, sem o consentimento expresso da IBM.

Exceto como expressamente concedido nesta permissão, nenhuma outra permissão, licença ou direito é concedido, expresso ou implícito, para as Publicações ou quaisquer informações, dados, software ou outra propriedade intelectual contida.

A IBM reserva-se o direito de retirar as permissões concedidas aqui sempre que, a seu critério, o uso das Publicações for prejudicial ao seu interesse ou, conforme determinado pela IBM, as instruções definidas anteriormente não estiverem sendo adequadamente seguidas.

O Cliente não poderá fazer download, exportar ou re-exportar estas informações exceto quando em conformidade total com todas as leis e regulamentações aplicáveis, incluindo todas as leis e regulamentações de exportação dos Estados Unidos.

A IBM NÃO FAZ QUALQUER TIPO DE GARANTIA QUANTO AO CONTEÚDO DESTAS PUBLICAÇÕES. AS PUBLICAÇÕES SÃO FORNECIDAS "NO ESTADO EM QUE SE ENCONTRAM", SEM GARANTIA DE NENHUM TIPO, SEJA EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS NÃO SE LIMITANDO ÀS GARANTIAS IMPLÍCITAS (OU CONDIÇÕES) DE NÃO-INFRAÇÃO, COMERCIALIZAÇÃO OU ADEQUAÇÃO A UM DETERMINADO PROPÓSITO.

Apêndice B. Avisos

Estas informações foram desenvolvidas para produtos e serviços oferecidos nos Estados Unidos. As informações sobre produtos não IBM baseiam-se nas informações disponíveis no momento da primeira publicação deste documento e estão sujeitas a mudanças.

É possível que a IBM não ofereça os produtos, serviços ou recursos discutidos nesta publicação em outros países. Consulte um representante IBM local para obter informações sobre produtos e serviços disponíveis atualmente em sua área. Qualquer referência a produtos, programas ou serviços IBM não significa que apenas produtos, programas ou serviços IBM possam ser utilizados. Qualquer produto, programa ou serviço funcionalmente equivalente, que não infrinja nenhum direito de propriedade intelectual da IBM poderá ser utilizado em substituição a este produto, programa ou serviço. Entretanto, a avaliação e verificação da operação de qualquer produto, programa ou serviço não IBM são de responsabilidade do Cliente.

A IBM pode ter patentes ou solicitações de patentes pendentes relativas a assuntos tratados nesta publicação. O fornecimento desta publicação não lhe garante direito algum sobre tais patentes. Pedidos de licença devem ser enviados, por escrito, para:

Gerência de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur 138-146
Botafogo
Rio de Janeiro - RJ
CEP 22290-240

Para pedidos de licença relacionados a informações de DBCS (Conjunto de Caracteres de Byte Duplo), entre em contato com o Departamento de Propriedade Intelectual da IBM em seu país ou envie pedidos de licença, por escrito, para:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

O parágrafo a seguir não se aplica a nenhum país em que tais disposições não estejam de acordo com a legislação local: A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO "NO ESTADO EM QUE SE ENCONTRA", SEM GARANTIA DE NENHUM TIPO, SEJA EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS A ELAS NÃO SE LIMITANDO, AS GARANTIAS IMPLÍCITAS DE NÃO INFRAÇÃO, COMERCIALIZAÇÃO OU ADEQUAÇÃO A UM DETERMINADO PROPÓSITO. Alguns países não permitem a exclusão de garantias expressas ou implícitas em certas transações; portanto, essa disposição pode não se aplicar ao Cliente.

Essas informações podem conter imprecisões técnicas ou erros tipográficos. São feitas alterações periódicas nas informações aqui contidas; tais alterações serão

incorporadas em futuras edições desta publicação. A IBM pode, a qualquer momento, aperfeiçoar e/ou alterar os produtos e/ou programas descritos nesta publicação, sem aviso prévio.

Quaisquer referências nestas informações a Web sites que não são de propriedade da IBM são fornecidas apenas para conveniência e não funcionam, de maneira nenhuma, como endosso a essas Web sites. Os materiais contidos nesses Web sites não fazem parte dos materiais desse produto IBM e a utilização desses Web sites é de inteira responsabilidade do Cliente.

A IBM pode utilizar ou distribuir as informações fornecidas da forma que julgar apropriada sem incorrer em qualquer obrigação para com o Cliente.

Licenciados deste programa que desejam obter informações sobre este assunto com objetivo de permitir: (i) a troca de informações entre programas criados independentemente e outros programas (incluindo este) e (ii) a utilização mútua das informações trocadas, devem entrar em contato com:

Gerência de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur, 138-146
Botafogo
Rio de Janeiro, RJ
CEP 22290-240

Tais informações podem estar disponíveis, sujeitas a termos e condições apropriadas, incluindo em alguns casos, o pagamento de uma taxa.

O programa licenciado descrito nesta publicação e todo o material licenciado disponível são fornecidos pela IBM sob os termos do Contrato com o Cliente IBM, do Contrato Internacional de Licença do Programa IBM ou de qualquer outro contrato equivalente.

Todos os dados de desempenho aqui contidos foram determinados em um ambiente controlado. Portanto, os resultados obtidos em outros ambientes operacionais podem variar significativamente. Algumas medidas podem ter sido tomadas em sistemas em nível de desenvolvimento e não há garantia de que estas medidas serão iguais em sistemas geralmente disponíveis. Além disso, algumas medidas podem ter sido estimadas por extrapolação. Os resultados reais podem variar. Os usuários deste documento devem verificar os dados aplicáveis para o seu ambiente específico.

As informações relativas a produtos não IBM foram obtidas junto aos fornecedores dos respectivos produtos, de seus anúncios publicados ou de outras fontes disponíveis publicamente. A IBM não testou estes produtos e não pode confirmar a precisão de seu desempenho, compatibilidade nem qualquer outra reivindicação relacionada a produtos não IBM. Dúvidas sobre os recursos de produtos não IBM devem ser encaminhadas diretamente a seus fornecedores.

Todas as declarações relacionadas aos objetivos e intenções futuras da IBM estão sujeitas a alterações ou cancelamento sem aviso prévio e representam apenas metas e objetivos.

Estas informações contêm exemplos de dados e relatórios utilizados nas operações diárias de negócios. Para ilustrá-los da forma mais completa possível, os exemplos

podem incluir nomes de indivíduos, empresas, marcas e produtos. Todos os nomes são fictícios e qualquer semelhança com nomes e endereços utilizados por uma empresa real é mera coincidência.

LICENÇA DE COPYRIGHT:

Estas informações contêm programas de aplicativos de amostra na linguagem fonte, ilustrando as técnicas de programação em diversas plataformas operacionais. O Cliente pode copiar, modificar e distribuir estes programas de amostra sem a necessidade de pagar à IBM, com objetivos de desenvolvimento, utilização, marketing ou distribuição de programas aplicativos em conformidade com a interface de programação de aplicativo para a plataforma operacional para a qual os programas de amostra são criados. Esses exemplos não foram testados completamente em todas as condições. Portanto, a IBM não pode garantir ou implicar a confiabilidade, manutenção ou função destes programas. Os programas de amostra são fornecidos "no estado em que se encontram", sem garantia de nenhum tipo. A IBM não poderá ser responsabilizada por qualquer dano causado pelo uso dos programas de amostra pelo Cliente.

Cada cópia ou parte destes programas de amostra ou qualquer trabalho derivado deve incluir um aviso de copyright com os dizeres:

© (*nome da empresa*) (*ano*). Partes deste código são derivadas dos Programas de Amostra da IBM Corp. © Copyright IBM Corp. *_digite o ano ou anos_*. Todos os direitos reservados.

Marcas Registradas

IBM, o logotipo IBM e ibm.com são marcas ou marcas registradas da International Business Machines Corp., registradas em vários países no mundo todo. Outros nomes de produtos e serviços podem ser marcas registradas da IBM ou de outras empresas. Uma lista atual de marcas registradas da IBM está disponível na web em “Copyright and trademark information” em www.ibm.com/legal/copytrade.shtml.

Os termos a seguir são marcas ou marcas registradas de outras empresas

- Linux é uma marca registrada de Linus Torvalds nos Estados Unidos e/ou em outros países.
- Java e todas as marcas registradas e logotipos baseados em Java são marcas ou marcas registradas da Oracle e/ou de suas afiliadas.
- UNIX é uma marca registrada do The Open Group nos Estados Unidos e em outros países.
- Intel, o logotipo Intel, Intel Inside, o logotipo Intel Inside, Celeron, Intel SpeedStep, Itanium e Pentium são marcas ou marcas registradas da Intel Corporation ou suas subsidiárias nos Estados Unidos e em outros países.
- Microsoft, Windows, Windows NT e o logotipo Windows são marcas registradas da Microsoft Corporation nos Estados Unidos e/ou em outros países.

Outros nomes de empresas, produtos ou serviços podem ser marcas registradas ou marcas de serviços de terceiros.

Índice Remissivo

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Impresso no Brasil

S517-0064-00

Spine information:

IBM DB2 10.1 para Linux, UNIX e Windows

Referência e Guia do Usuário do Spatial Extender