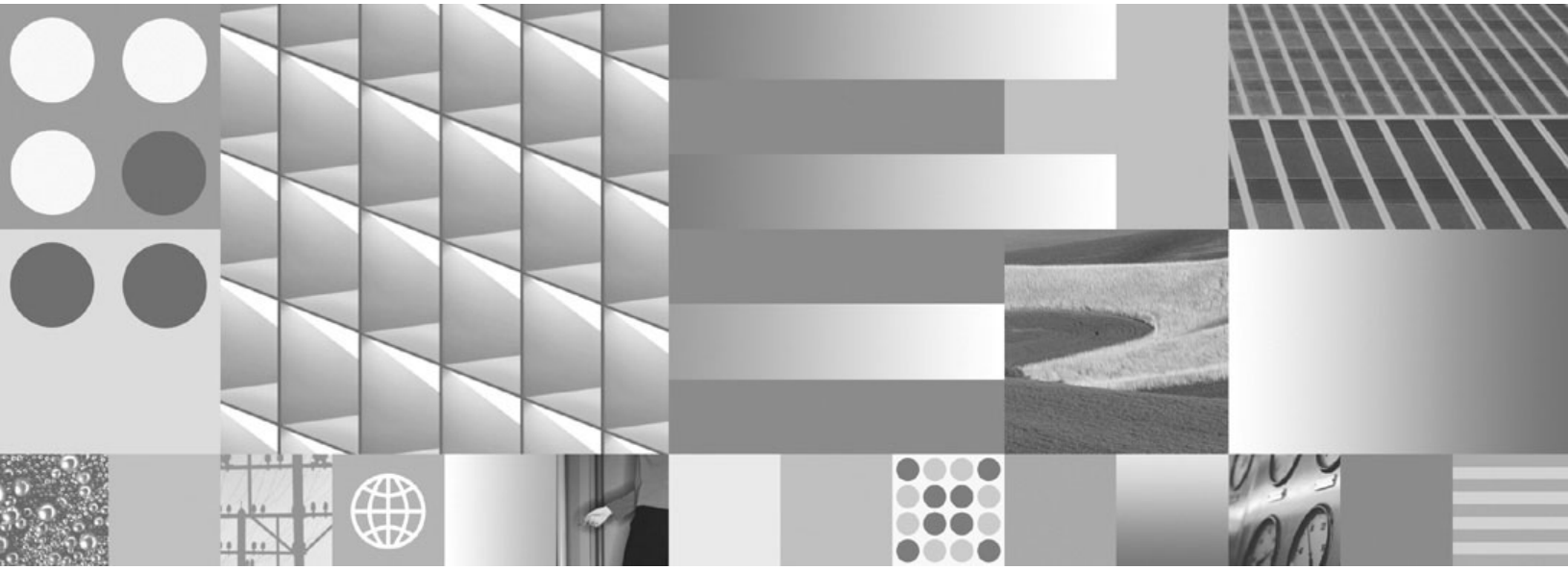
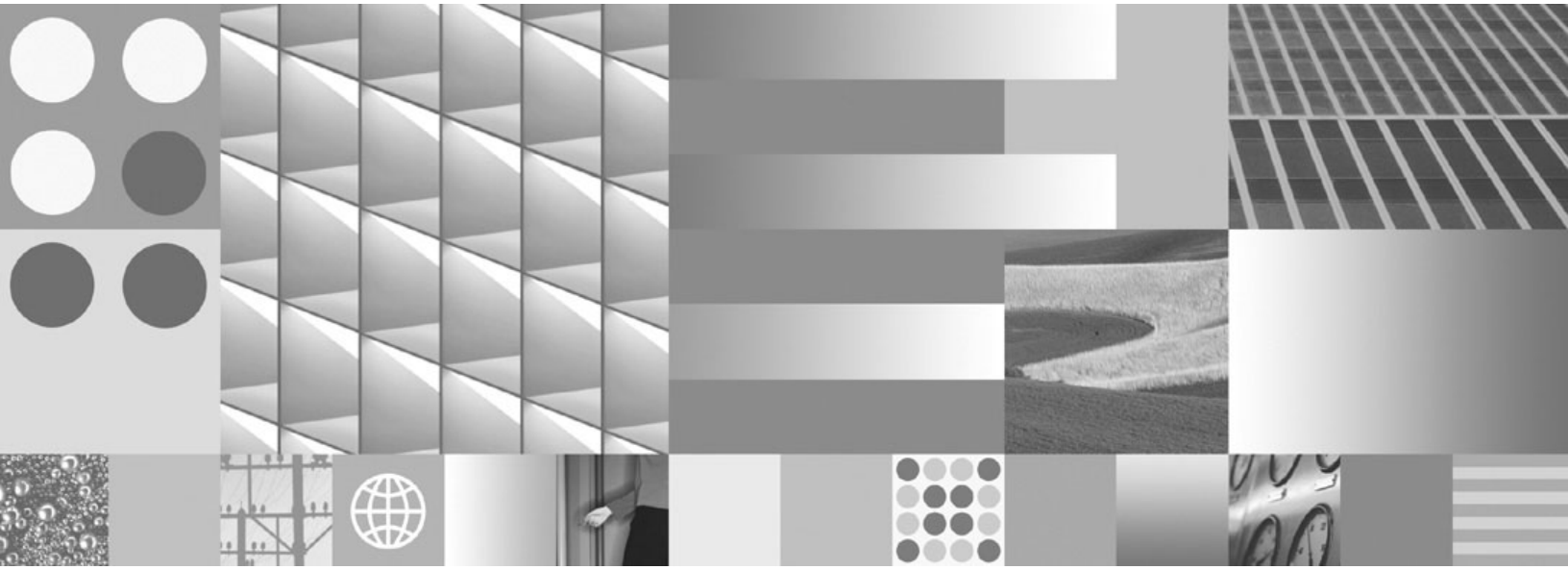
IBM DB2 9.7
for Linux, UNIX, and Windows



Spatial Extender および Geodetic Data Management Feature ユーザーズ・ガイドおよびリファレンス

SC88-5900-00
(英文原典：SC27-2468-00)

IBM DB2 9.7
for Linux, UNIX, and Windows



Spatial Extender および Geodetic Data Management Feature ユーザーズ・ガイドおよびリファレンス

SC88-5900-00
(英文原典：SC27-2468-00)

ご注意

本書および本書で紹介する製品をご使用になる前に、523 ページの『付録 B. 特記事項』に記載されている情報をお読みください。

本書には、IBM の専有情報が含まれています。その情報は、使用許諾条件に基づき提供され、著作権により保護されています。本書に記載される情報には、いかなる製品の保証も含まれていません。また、本書で提供されるいかなる記述も、製品保証として解釈すべきではありません。

IBM 資料は、オンラインでご注文いただくことも、ご自分の国または地域の IBM 担当員を通してお求めいただくこともできます。

- オンラインで資料を注文するには、www.ibm.com/shop/publications/order にある IBM Publications Center をご利用ください。
- ご自分の国または地域の IBM 担当員を見つけるには、www.ibm.com/planetwide にある IBM Directory of Worldwide Contacts をお調べください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： SC27-2468-00
IBM DB2 9.7
for Linux, UNIX, and Windows
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2009.7

© Copyright International Business Machines Corporation 1998, 2009.

目次

© Copyright IBM Corp. 1998, 2009

第 1 章 DB2 Spatial Extender の概要

このセクションでは、DB2® Spatial Extender の目的およびサポートされるデータを説明し、基礎となる概念をどのように組み合わせているかを説明することで、当製品の概要を紹介しています。

DB2 Spatial Extender の目的

地理的特徴に関する空間情報の生成や分析を行い、この情報の基になるデータの保管や管理を行うには、DB2® Spatial Extender を使用します。地理的特徴 (ここでの説明では、短く、地形 と呼ぶこともある) は、現実の世界で、識別可能なロケーションをもつなんらかのものであるか、あるいは、識別可能なロケーションに存在していると考えられるなんらかのものです。地形は、以下のような形として存在します。

- オブジェクト (つまり、あらゆる種類の具体的エンティティー); 例えば、河川や森、山岳地帯など。
- スペース; 例えば、危険な地域の周りの安全ゾーン、特定の企業がサービスを提供する販売エリアなど。
- 定義可能なロケーションで発生するイベント; 例えば、特定の交差点で発生した交通事故、または、特定の店での販売取引など。

地形はさまざまな環境に存在します。例えば、上でとりあげた、河川、森、山岳地帯などのオブジェクトは、自然環境に属します。その他の、街、ビルディング、オフィスなどのオブジェクトは、文化環境に属します。さらに、公園、動物園、農場などのその他のオブジェクトは、自然環境と文化環境を組み合わせたものです。

ここでの説明では、空間情報 とは、DB2 Spatial Extender によって、ユーザーが使用できるようになる、ある種の情報を意味します。つまり、地形のロケーションに関する正確な情報のことです。空間情報の例を以下に示します。

- 地図上の地形のロケーション (例えば、街の位置を定義する経度や緯度の値)
- 互いの位置と比較した相対的な地形の位置 (例えば、ある街で病院や診療所が存在する地点や局地的な地震発生ゾーンに近接する街の居住地など)。
- 地形同士の関係 (例えば、特定の地域内を流れる特定の水系に関する情報や、その地域内のその水系の支流に架かっている橋に関する情報など)。
- 1 つまたは複数の地形に適用される測定値 (例えば、オフィスビルからその敷地の境界までの間の距離や、野鳥保護区域の周囲の長さ)

空間情報を、単独で、あるいは従来のリレーショナル・データと組み合わせて使用すると、サービスを提供するエリアを定義したり、マーケットにできる可能性のある場所を決定するなどの作業を行う際に役立ちます。例えば、自治体の福祉管理者が、福祉援助の申請者と受取人が実際に住んでいる行政区内の地域を検証する必要があるとします。DB2 Spatial Extender を使用すると、行政区内の地域の位置と、申請者と受取人の住所からこの情報が導き出されます。

© Copyright IBM Corp. 1998, 2009

次に、レストラン・チェーンのオーナーが、近隣の街に事業を展開したいと思っているとします。新しい店を開く場所を決定するには、「これらの街の中のどの場所ならば、自分の店を頻繁に利用する常連客が集まるだろうか? 主要な高速道路はどこにあるか? 犯罪発生率が低いのはどこか? 競争相手のレストランがある場所はどこか?」という質問の答えを出す必要があります。 DB2 Spatial Extender と DB2 は、これらの質問に答えるための情報を生成することができます。さらに、フロントエンド・ツールも、必須のものではありませんが、一部の情報を生成することができます。説明するために、視覚化ツールは、DB2 Spatial Extender が生成する、例えば、常連客が集まる場所や計画しているレストランに近い主要な高速道路などの情報を、地図上の図形にして表すことができます。ビジネス・インテリジェンス・ツールは、競合レストランの名前や説明などの関連情報を、レポート形式で表すことができます。

データによって地形を表す方法

DB2® Spatial Extender では、地形は、表の行にあるデータ項目などの、1 つ以上のデータ項目によって表すことができます。 (データ項目 は、リレーショナル表のセルを占有する値 (1 つ以上) です。)例として、オフィスビルと居住地について考えてみます。図 1 で、BRANCHES 表の個々の行は銀行の支店を表します。同様に、図 1 の CUSTOMERS 表の個々の行は、銀行の顧客を表します。ただし、個々の行のサブセット、特に、顧客の住所を構成するデータ項目が、顧客の居住地を表していると見なすことができます。

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

図 1. 地形を表すデータ： BRANCHES 表のデータの行は銀行の支店を表している。CUSTOMERS 表の住所データは、顧客の居住地を表している。2 つの表の名前と住所は架空のもの。

図 1 の表には、銀行の支店と顧客を識別し、記述したデータがあります。以下では、ビジネス・データ などのデータについて説明します。

支店の住所と顧客の住所を示す値である、ビジネス・データのサブセットは、空間情報を生成する値に変換できます。例えば、図 1 では、支店の住所が 92467 Airzone Blvd., San Jose, CA 95141, USA. になっています。顧客の住所は 9 Concourt Circle, San Jose, CA 95141, USA. です。DB2 Spatial Extender は、これらの住所を、支店と顧客の家のある、互いの場所と比較した、相対的な場所として示す値に変換できます。 3 ページの図 2 は、このような値を入れることを指定し、列を新たにした BRANCHES 表と CUSTOMERS 表を示しています。

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

図 2. 空間列が追加された表： 個々の表の LOCATION 列には、住所に対応した座標が入れられます。

空間情報は、LOCATION 列に保管されているデータ項目から導き出されるので、ここでの説明では、これらのデータ項目を空間データと呼ぶことにします。

空間データの性質

空間データはロケーションを識別する座標から構成されています。Spatial Extender は、X と Y、または経度と緯度の値で指定される 2 ディメンション座標で動作します。

座標 は、以下のいずれかを示す数値です。

- 軸上での、原点から見た相対的な位置。長さの単位で表される。
- ある線、あるいは面を基準にした相対的な方向。角度の単位で表される。

例えば、緯度は、赤道面を基準にした相対的な角度を表す座標で、通常、単位は「度」になります。経度は、グリニッジ子午線を基準にした相対的な角度を表す座標で、これも通常、単位は「度」になります。したがって、イエローストーン国立公園の位置は、地図上では、赤道を基準にした北緯 44.45 度という緯度と、グリニッジ子午線を基準にした西経 110.40 度という経度で表されます。厳密には、これらの座標は、米国のイエローストーン国立公園の中心を示しています。

緯度、経度、その基準ポイント、基準線、基準面、測定単位、および他の関連したパラメーターの定義は、まとめて座標システム といわれます。緯度と経度以外の値に基づく座標システムもあります。こうした座標システムには、独自の基準ポイント、基準線、基準面、測定単位、およびその他の識別パラメーター (投影トランスフォーメーションなど) があります。

最も単純な空間データ項目は、1 つの地理的な位置を定義する 2 つの座標から成り立ちます。これより複雑な空間データ項目は、道路や河川などの線状の通り道を定義する複数の座標から成り立ちます。さらに複雑な項目は、一区画の土地や洪水地帯の境界など、領域の境界を定義する座標から成り立ちます。

個々の空間データ項目は、空間データ・タイプのインスタンスです。単一のロケーションを示す座標のデータ・タイプは ST_Point です。線状の道を定義する座標のデータ・タイプは ST_LineString です。領域の境界を定義する座標のデータ・タイプは ST_Polygon です。これらのタイプと、他の空間データ・タイプは、単一の階層に属する構造タイプです。

測地データの性質

測地データは、緯度、経度といった座標を使用した空間データの一種です。測地データに使用される座標システムは、球状で、連続していて、しかも閉じているという面の上での位置を表すためのものです。

DB2 Geodetic Data Management Feature は、Spatial Extender と同じデータ・タイプと関数を使用して、DB2 データベースに測地データを保管します。地球を平面の地図で表現する Spatial Extender とは異なり、Geodetic Data Management Feature は、極地や日付変更線などで途切れたり、継ぎ合わされたりしない連続した球体として地球を扱います。平面の地図は、球面座標を平面座標にトランスフォームするために投影座標を必要とします。それに対し、Geodetic Data Management Feature は地球表面の楕円モデル上の緯度および経度を使用します。線の交差、領域の重なり、距離、面積など計算の正確さ、精度は、位置に関係なく同じです。

空間データのソース

空間データを取得するには、以下のようにします。

- ビジネス・データから導出する
- 空間処理関数から生成する
- 外部ソースからインポートする

ビジネス・データをソース・データとして使用する

DB2 Spatial Extender では、空間データを、住所などのビジネス・データから導出できます (2 ページの『データによって地形を表す方法』 を参照)。 このプロセスをジオコーディング といいます。関係する一連の事柄について考慮するために、 3 ページの図 2 を『ジオコーディング実行前』のピクチャー、図 3 を『ジオコーディング実行後』のピクチャーとします。 3 ページの図 2 の BRANCHES と CUSTOMERS の両方の表には、空間データ用に指定された列があります。これらの表の中の住所は DB2 Spatial Extender によりジオコーディングされて、住所に対応した座標が求められ、その座標が表の列に入れられるとします。図 3 はこの結果を示しています。

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

図 3. ソース・データから導出された空間データを含む表： CUSTOMERS 表の LOCATION 列には、ADDRESS、CITY、POSTAL CODE、STATE_PROV、および COUNTRY の各列の住所から導出された座標が入っている。同様に、BRANCHES 表の LOCATION 列には、この表の ADDRESS、CITY、POSTAL CODE、STATE_PROV、および COUNTRY の各列の住所から導出された座標が入っている。

DB2 Spatial Extender は、ジオコーダー と呼ばれる関数を使用して、ビジネス・データを座標に変換し、空間処理関数がそのデータに対して操作を行えるようにします。

空間データを生成する関数を使用する

関数を使用して入力データから空間データを生成できます。

空間データは、ジオコーダーだけでなく、その他の関数によっても、生成することができます。例えば、支店が BRANCHES 表に定義されている銀行が、個々の支店から半径 5 マイル内に住んでいる顧客の数を知りたいとします。この銀行がデータベースからこの情報を取得するには、その前に、個々の支店のまわりの、指定半径内にある領域を定義しなければなりません。この種の定義は、DB2 Spatial Extender の関数 ST_Buffer を使用して作成できます。ST_Buffer により、個々の支店の座標を入力として使用して、その領域の境界線を確定する座標を生成できます。図 4 は、ST_Buffer によって BRANCHES 表に情報が入れられた様子を示しています。

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

図 4. 既存の空間データから導出された新しい空間データを含む表： SALES_AREA 列内の座標は、ST_Buffer によって LOCATION 列内の座標から導出されたもの。SALES_AREA 列内の座標は、LOCATION 列内の座標と同じくシミュレーションであり、実在しない。

DB2 Spatial Extender には、 ST_Buffer 以外にも、既存の空間データから新しく空間データを導出する関数があります。

空間データのインポート

Spatial Extender は、シェイプ・ファイル・フォーマットの空間データをインポートするためのサービスを提供しています。

シェイプ・ファイル・フォーマットの空間データは、インターネットを介して多くのソースから入手できます。http://www.ibm.com/software/data/spatial/db2spatial にあるトライアルとデモの Web ページで DB2 Spatial Extender Sample Map Data オファリングを選択することにより、国、州、市、川など、米国および世界中の地形のデータおよび地図をダウンロードできます。

外部データ・ソースによって提供されるファイルから空間データをインポートできます。 この種のファイルには、通常、地図に適用されるデータが入っています。例えば、道路網、洪水地帯、地震断層などです。この種のデータと、作成した空間データを組み合わせて使用すると、利用可能な空間情報を増やすことができます。例えば、公共事業を行う省庁が、ある住宅地がどんな災害に遭いやすいか判別する場合に、ST_Buffer を使用してその住宅地を含む領域を定義します。その後に、洪水地帯や断層に関するデータをインポートして、災害を起こしそうな区域のうちどれが、先に定義した領域と重なり合っているかを調べることができます。

地形、空間情報、空間データおよび形状の組み合わせ方

このセクションでは、DB2® Spatial Extender の操作の基礎となるいくつかの基本的概念、すなわち、地理的特徴、空間情報、空間データ、および形状について要約しています。

DB2 Spatial Extender を使用すると、地理的に定義できるもの、すなわち、地球上でのロケーション、または地球上のある地域内で定義できるものに関する正確な情報を入手できます。 DB2 ドキュメントでは、このような正確な情報を*空間情報* と呼び、定義できるものを*地理的特徴* (ここでは短く*地形*) と呼びます。

例えば、DB2 Spatial Extender を使用すると、ゴミの埋立地として提案されている地域がどこかの住居地域と重ならないか調べることができます。ここで、住居地域と提案区域は、地形です。重なり合う部分があるかどうかは、空間情報の例となります。重なると判別された場合、重なり合う範囲も空間情報の例の 1 つです。

空間情報を作成するには、DB2 Spatial Extender は、地形のロケーションを定義するデータを処理する必要があります。*空間データ* と呼ばれるこのようなデータは、地図または同様な図法でのロケーションを示す座標から構成されます。例えば、ある地形が他の地形に重なっているかどうかを判別するには、DB2 Spatial Extender は、一方の地形の座標位置を、もう一方の地形の座標との関係で判別する必要があります。

空間情報テクノロジーの世界では、地形は*形状 (geometry)* と呼ばれるシンボルで表現されたものと考えるのが一般的です。形状は、一部は視覚的に表現され、一部は数学的に表現されます。まず、その視覚的な部分について考えましょう。公園や街など、間口と奥行きを持つある地形をシンボルで表すと、複数の辺をもつ図形になります。このような形状を*ポリゴン (多角形)* と呼びます。川や道路などの線状の地形は、シンボルで表すと線になります。このような形状を*折れ線* と呼びます。

形状は、それが表す実際の地形に対応するプロパティーを持ちます。これらのプロパティーのほとんどは、数学的に表現できます。例えば、地形の座標は、それらをまとめて、地形に対応する形状のプロパティーの 1 つを構成しています。もう 1 つのプロパティーはディメンション と呼ばれ、地形が長さまたは幅を持つかどうかを示す数値です。

空間データとある種の空間情報は、形状の観点から見ることができます。前に述べた、住居地域とゴミ埋立地提案区域の例で考えてみましょう。住居地域に関する空間データには、DB2 データベースの表の列に格納されている座標が含まれます。慣例により、格納されているものは、単なるデータとしてではなく、実際の形状と見なされます。住居地域には間口と奥行きがあるので、形状はポリゴンだと分かります。

空間データと同様に、ある種の空間情報も形状として見られます。例えば、住居地域がゴミ埋立地として提案された区域と重なっているかどうかを判別するには、DB2 Spatial Extender は、その埋立地をシンボル化したポリゴンの座標を、住居地域を表すポリゴンの座標と比べる必要があります。その結果として得られる情報 (オーバーラップする区域) 自体も、ポリゴン、すなわち、座標、ディメンション、その他のプロパティーを持つ形状と見なされます。

第 2 章 形状について

この章では、形状と呼ばれる情報のエンティティーを説明します。形状は座標からなり、地理上の地形を表します。以下のトピックを取り上げます。

- 形状
- 形状のプロパティー

形状

Webster の Revised Unabridged Dictionary によれば、*geometry (幾何学)* とは、『数学の 1 分野であり、実線、曲面、輪郭、および角度について、その関係、特質、計測を研究するもの、大きさのプロパティーと関係を扱う科学、空間の関係を扱う科学』と定義されています。また、形状 (geometry) という単語には地形の意味もあり、過去数千年にわたって、世界の地図を作成するために地図製作者が使用してきました。「形状」というこの新しい意味の抽象的な定義は、「地上の 1 つの地形をあらわす 1 つのポイントまたはポイントの集約」です。

DB2 Spatial Extender では、形状の操作上の定義は『地形のモデル』です。 このモデルは、地形の座標を使って表現できます。このモデルは情報を伝達します。例えば、座標は、固定された基準ポイントとの相対的な関係で地形の位置を示します。また、このモデルは、情報を生成するためにも使用できます。例えば、ST_Overlaps 関数は、入力として 2 つの近接領域の座標を受け取り、その領域が重なるかどうかの情報を戻すことができます。

ある形状が表す地形の座標は、その形状のプロパティー と見なされます。形状の種類によっては、他のプロパティー、例えば、区域、長さ、境界などを持つことがあります。

DB2 Spatial Extender がサポートする形状は、以下に示すような階層を形成します。この形状階層は、OpenGIS Consortium, Inc. (OGC) の資料である「OpenGIS Simple Features Specification for SQL」に定義されています。インスタンス化可能な階層のメンバーは 7 つです。それぞれに特定の座標値によって定義でき、図に示すようなかたちでレンダリングし、視覚化できます。

© Copyright IBM Corp. 1998, 2009

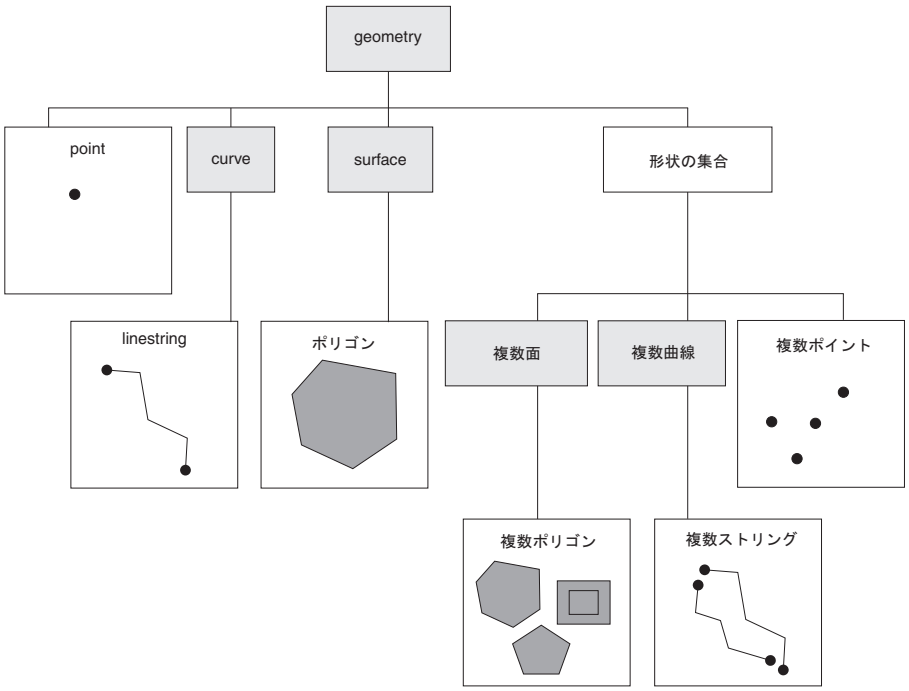


図 5. DB2 Spatial Extender のサポートする形状の階層： この図のインスタンス化可能形状には、形状がどのように可視的に描かれるかの例も含まれています。

DB2 Spatial Extender がサポートする空間データは、この図に示してある形状で示されているものです。

この図が示すように、形状 という名前のスーパークラスは、階層のルートです。階層内のルート・タイプおよびその他の固有のサブタイプは、インスタンス化できません。また、ユーザーは、インスタンス化できる、またはできない独自のサブタイプを定義できます。

そのサブタイプは、基本形状サブタイプと同種集合サブタイプの 2 つのカテゴリーに分かれます。

基本形状には、以下のものがあります。

ポイント
1 つのポイントです。ポイントは、座標上の東西の線 (緯線など) が南北の線 (経線など) と交差する交点に位置するものとして認識される個々の地形を表します。例えば、世界地図上で、個々の都市が緯線と経線の交点に位置する場合の表記を考えてください。この場合、ポイントは各都市を表すことができます。

折れ線 2 つ以上のポイント間の線です。直線とは限りません。折れ線は、線状の地形 (道路、運河、パイプラインなど) を表します。

ポリゴン
多角形、あるいは多角形の中の面です。ポリゴンは、複数の辺からなる地形 (森、野生生物の生息地など) を表します。

同種集合には、以下のものがあります。

複数ポイント
複数のポイント形状の集合です。複数ポイントは、それぞれが東西に走る座標軸 (緯線など) と南北に走る座標軸 (経線など) の交点にある複数の地形部分から成り立つ地形を表します (例えば、それぞれの島が緯線と経線の交点にある群島)。

複数折れ線
複数折れ線を持つ、複数の曲線形状の集合です。複数折れ線は、複数の地形部分から構成される地形 (例えば、水系や高速道路網) を表します。

複数ポリゴン
複数ポリゴンを持つ、複数の面の形状の集合です。複数ポリゴンは、複数の辺またはコンポーネントから成る、複数部分を持つ地形 (例えば、特定地域内の集団農場または湖水系) を表します。

同種集合は、それらの名前が示しているように、基本形状の集合です。同種集合は、基本形状のプロパティーを共有するほかに、それぞれのプロパティーもいくつか持っています。

形状のプロパティー

ここでは、形状のプロパティーについて説明します。プロパティーには、以下のものがあります。

- 形状が属しているタイプ
- 形状の座標
- 形状の内部、境界、外部
- 単純であるか、非単純であるかの特性
- 空であるか、空でないかの特性
- 形状の最小外接長方形またはエンベロープ
- ディメンション
- 形状に関連付けられる空間参照系の ID

タイプ

各形状は、DB2 Spatial Extender によってサポートされる形状の階層中のタイプに属します。階層にある 7 つのタイプ (ポイント、折れ線、ポリゴン、形状の集合、複数ポイント、複数折れ線、および複数ポリゴン) は特定の座標値によって定義できます。

形状の座標

すべての形状には、少なくとも 1 つの X 座標と 1 つの Y 座標が含まれます。ただし、空の形状には、座標はまったく含まれません。また、形状には、1 つ以上の Z 座標と M 座標が含まれることがあります。X、Y、Z および M の座標は、倍精度数として表されます。以下のサブセクションでは、次のことについて説明します。

- X 座標と Y 座標
- Z 座標

- M 座標

X 座標と Y 座標

X 座標値 は、東または西の基準ポイントからの相対的なロケーションを示します。
Y 座標値 は、北または南の基準ポイントからの相対ロケーションを示します。

Z 座標

形状の中には、高さまたは深度が関連付けられるものがあります。地形の形状を形成する各ポイントは、オプションとして、地表からの高度、または、深度を持つことができます。

M 座標

M 座標 (ものさし) は、地形についての情報を伝達する値であり、地形のロケーションを定義する座標と一緒に格納されます。例えば、アプリケーションで高速道路を示したいとします。使用するアプリケーションで、直線距離またはマイル標を示す値を処理したい場合は、これらの値を、高速道路沿いの地点を定義する座標とともに格納することができます。M 座標は、倍精度数として表されます。

内部、境界、および外部

すべての形状は、その内部、境界、外部によって定義されるスペース内の位置を占めます。形状の外部とは、その形状が占めないすべてのスペースです。形状の境界は、その内部と外部のインターフェースの役割を果たしています。内部は、その形状が占めるスペース部分です。

単純か非単純か

ある種の形状サブタイプ (折れ線、複数ポイント、および複数折れ線) の値は、単純または非単純のいずれかです。形状がそのサブタイプに課せられたすべてのトポロジー規則に従っている場合は、形状は単純であり、従っていなければ非単純です。折れ線は、その内部と交差していなければ、単純です。複数ポイントは、そのエレメントがどれも同じ座標スペースを占めていなければ、単純です。ポイント、面、複数面、および空の形状は、常に単純です。

閉じた曲線

曲線は、開始ポイントと終了ポイントが等しい場合、閉じていることになります。複数曲線は、そのエレメントのすべてが閉じていれば、閉じている、ということになります。リングは、閉じている曲線の中でも特に単純なものです。

空か空でないか

形状がその中にポイントを 1 つも含んでいなければ、形状は空です。空の形状のエンベロープ、境界、内部および外部は定義されておらず、NULL として表されます。空の形状は、常に単純です。空のポリゴンと複数ポリゴンは、0 の区域を持ちます。

最小外接長方形 (MBR)

形状の MBR は、最小と最大の (X,Y) 座標で形成される外接形状です。以下の特殊な場合を除いて、形状の MBR は、外接長方形を形成します。

- 最小と最大の X 座標が同じで、最小と最大の Y 座標も同じであるため、どのポイントの MBR も、ポイントそのものである場合
- 水平の折れ線または垂直の折れ線の MBR が、元の折れ線の境界 (終了ポイント) が表す折れ線である場合

ディメンション

形状は、–1、0、1 または 2 のディメンションを持つことができます。各ディメンションは以下のようになります。

–1 空

0 長さがなく、区域は 0

1 0 より大きな長さを持ち、区域は 0

2 0 より大きな区域を持つ

ポイント・サブタイプと複数ポイント・サブタイプのディメンションは 0 です。ポイントは、1 組の座標でモデル化できるディメンション地形を表し、一方、複数ポイント・サブタイプは、1 組のポイントでモデル化する必要のあるデータを表します。

折れ線サブタイプと複数折れ線サブタイプのディメンションは 1 です。これによって、道路の一部、支流を持つ水系、および、もともとが線状であるその他の地形を格納します。

ポリゴン・サブタイプと複数ポリゴン・サブタイプのディメンションは 2 です。その境界線が定義可能な区域、すなわち、森、土地の一部、湖などを囲んでいる地形は、ポリゴンまたは複数ポリゴンのいずれかのデータ・タイプによって示すことができます。

空間参照系の ID

空間参照系の数字 ID によって、どの空間参照系を使用して形状を表現するかが決められます。

データベースが認識するすべての空間参照系は、
DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビューによってアクセスできます。

第 3 章 DB2 Spatial Extender の使用法

DB2 Spatial Extender の使用法

DB2® Spatial Extender のサポートと使用法には、DB2 Spatial Extender のセットアップと空間データを使用するプロジェクトの組み立てという、2 つの主な作業が含まれます。このトピックでは、空間タスクの実行に使用できるインターフェースを紹介します。

DB2 Spatial Extender および関連機能に対するインターフェース

DB2 Spatial Extender をセットアップし、空間データを使用するプロジェクトを作成するために、いくつかのインターフェースがあります。 このようなインターフェースには、次のものがあります。

- DB2 Spatial および Geodetic Data Management Features のサポートを含むオープン・ソース・プロジェクト。 このサポートを含むオープン・ソース・プロジェクトには以下のようなものがあります。
  - GeoTools (http://www.geotools.org/)、空間アプリケーションを作成するための Java™ ライブラリー
  - GeoServer (http://docs.codehaus.org/display/GEOS/Home)、Web マップ・サーバーと Web 地形サーバー
  - uDIG (http://udig.refractions.net/confluence/display/UDIG/Home)、デスクトップの空間データ視覚化および分析アプリケーション
- DB2 コントロール・センター。これは DB2 Spatial Extender をサポートするウィンドウ、ノートブック、およびメニュー選択を含むグラフィカル・ユーザー・インターフェースです。
- DB2 Spatial Extender が提供するコマンド行プロセッサー (CLP)。 db2se CLP と呼ばれます。
- DB2 Spatial Extender のストアード・プロシージャーを呼び出すアプリケーション・プログラム。

空間を生成するために、その他のインターフェースがあります。それらのインターフェースは、以下のとおりです。

- DB2 CLP、DB2 コントロール・センターの照会ウィンドウ、またはアプリケーション・プログラムからサブミットする SQL 照会。
- 空間をグラフィカルに表現する視覚化ツール。この例としては、IBM® 向けに Environmental Systems Research Institute (ESRI) が作成した ArcExplorer for DB2 があります。 ArcExplorer for DB2 は、 DB2 Spatial Extender の下記の Web サイトからダウンロードすることができます。http://www.ibm.com/software/data/spatial/

© Copyright IBM Corp. 1998, 2009

DB2 Spatial Extender のセットアップとプロジェクトの作成のために行う作業

ここでは、DB2 Spatial Extender のセットアップと、空間データを使用するプロジェクトの組み立てに必要なタスクを簡単に説明します。 説明には、タスクを例示するシナリオが含まれています。タスクは 2 つのカテゴリーに分類されます。

- DB2 Spatial Extender のセットアップ
- 空間データを使用するプロジェクトの作成

DB2 Spatial Extender のセットアップ

ここでは、DB2 Spatial Extender をセットアップするために行うタスクをとりあげ、シナリオを使用して、架空の会社がそれぞれのタスクをどのように行うかを例をあげて示します。

DB2 Spatial Extender をセットアップするには、次のようにします。

1. 計画と準備を行います (どんなプロジェクトを作成するかを決める、どんなインターフェースを使用するかを決める、DB2 Spatial Extender の管理とプロジェクトの作成を行う人を選択するなど)。

   シナリオ: Safe Harbor 不動産保険会社の情報システム環境には、DB2 データベース・システムと空間データ専用の別個のファイル・システムが組み込まれています。 照会結果には、ある程度、両方のシステムのデータを組み合わせたデータが含まれます。例えば、DB2 表には収益に関する情報が格納され、ファイル・システムのファイルにはこの会社の支店のロケーションが格納されているとします。 こうすると、指定した額の収益をあげた支店を探索し、そのロケーションを特定することができます。しかし、2 つのシステムからのデータを統合することはできません (例えば、DB2 の列とファイル・システムのレコードを結合することはできません。また、照会の最適化などの DB2 サービスはファイル・システムでは使用できません)。この欠点を解消するために、Safe Harbor 社は DB2 Spatial Extender を購入して、新しく空間開発部門 (略して、空間部門と呼ぶ) を立ち上げます。

   空間部門の最初の仕事は、以下に示すように、DB2 Spatial Extender を Safe Harbor 社の DB2 環境に組み込むことです。

   - この部門の管理チームが、DB2 Spatial Extender のインストールとインプリメントを行う空間管理チームと、空間情報の生成と分析を行う空間情報分析チームを指名します。
   - 管理チームは UNIX® の基本的な知識があるので、DB2 Spatial Extender の管理に db2se CLP を使用することを決定します。
   - Safe Harbor 社の業務上の決定は主に顧客の要件に基づいて行われるので、管理チームは、顧客に関する情報を持っているデータベースに DB2 Spatial Extender をインストールすることを決定します。 この情報の大部分は CUSTOMERS という表に格納されます。

2. DB2 Spatial Extender をインストールします。

   シナリオ: 空間管理チームは、DB2 環境にある UNIX® マシンに DB2 Spatial Extender をインストールします。

3. DB2 Spatial Extender バージョン 8 をもっている場合は、空間データを DB2 のバージョン 9.5 に移行します。

   シナリオ: Safe Harbor 社が初めて購入したリリースは、バージョン 9.5 なので、マイグレーションの必要はありません。
4. 空間データを収容できるようにデータベースを構成します。空間処理関数、ログ・ファイル、DB2 Spatial Extender アプリケーション用として、データベースが、十分なメモリーとスペースを確保できるように、構成パラメーターを調整します。

   シナリオ: 空間管理チームのメンバーは、トランザクション・ログ特性、アプリケーション・ヒープ・サイズ、およびアプリケーション制御ヒープ・サイズを、DB2 Spatial Extender の要件に適合する値に調整します。
5. データベース用に空間リソースをセットアップします。これらのリソースには、システム・カタログ、空間データ・タイプ、空間処理関数、ジオコーダー、その他のオブジェクトなどがあります。これらのリソースのセットアップ・タスクのことを、空間操作のための、データベースの使用可能化といいます。

   DB2 Spatial Extender で提供されるジオコーダーでは、米国のアドレスが空間データに変換されます。 このジオコーダーは、DB2SE_USA_GEOCODER と呼ばれます。米国内外のアドレスや他の種類のデータを空間データに変換するジオコーダーは、貴社または他社が用意します。

   シナリオ: 空間管理チームが、計画中のプロジェクトに必要なリソースをセットアップします。
   - チームのメンバーが、空間操作にデータベースを使用できるように、コマンドを出してリソースを確保します。この種のリソースには、DB2 Spatial Extender カタログ、空間データ・タイプ、空間処理関数などがあります。
   - Safe Harbor 社は事業をカナダに展開し始めているので、空間管理チームは、カナダのアドレスを空間データに変換するためのジオコーダーの入手について、カナダのベンダーと交渉を始めます。 Safe Harbor 社は、まだ 2、3 カ月はそのようなジオコーダーを購入する予定ではありません。したがって、Safe Harbor 社が最初にデータを収集するロケーションは、米国になります。

空間データを使用するプロジェクトの作成

DB2 Spatial Extender のセットアップが終われば、空間データを使用するプロジェクトを開始することができます。 ここでは、そのようなプロジェクトの作成に関連したタスクをとりあげます。また、ビジネス・データと空間データを統合するために、Safe Harbor 不動産保険会社が行うシナリオを引き続き説明します。

空間データを使用するプロジェクトを作成するには、次のようにします。

1. 計画と準備を行います (プロジェクトの目標の設定、必要な表とデータの決定、使用する座標システムの決定など)。

   シナリオ: 空間部門は、プロジェクトを開発する準備をします。たとえば次のようにします。
   - 管理チームが、プロジェクトの目標を次のように設定します。
     - 新しい支店を設立する場所を決定します。

- 危険地域 (交通事故の発生率が高い地域、犯罪発生率が高い地域、洪水地帯、地震断層など) と顧客との近接の度合いに応じて、保険料を調整します。

- このプロジェクトは、米国の顧客とオフィスを考慮します。したがって、空間管理チームは以下のことを決定します。
  - DB2 Spatial Extender が提供する、米国用の座標システムを使用します。この座標システムは GCS_NORTH_AMERICAN_1983 と呼ばれます。
  - DB2SE_USA_GEOCODER は、米国のアドレスをジオコーディングするよう設計されているので、これを使用します。
- 空間管理チームは、プロジェクトの目標を満たすのに必要なデータと、このデータを入れる表を決定します。

2. 必要な場合は、座標システムを作成します。

   シナリオ: Safe Harbor 社は GCS_NORTH_AMERICAN_1983 を使用することにしているので、この会社はこのステップを無視することができます。

3. 既存の空間参照系が、希望する要求を満たすかどうかを判断します。要求を満たすシステムがない場合は、システムを作成します。

   空間参照系 は、パラメーター値のセットであり、以下のものが含まれます。

   - 与えられた座標の範囲により参照されるスペースの、可能な最大範囲を定義する座標。使用する座標システムから決定できる座標の可能な最大範囲を決定し、この範囲を反映する空間参照系を選択または作成する必要があります。
   - 座標の導出元となっている座標システムの名前。
   - 入力として受け取られた座標を、最も効率良く処理できる値に変換する、数学演算で使用される数。座標は、変換されたフォーマットで保管され、ユーザーには、元のフォーマットで戻されます。

   シナリオ: DB2 Spatial Extender には、GCS_NORTH_AMERICAN_1983 と一緒に使用するように設計された、空間参照系 NAD83_SRS_1 が用意されています。空間管理チームは、NAD83_SRS_1 の使用を決定します。

4. 必要に応じて、空間列を作成します。視覚化ツールが空間列のデータを読み取るときは、多くの場合、その列は、列が属する表やビュー内の唯一の空間列でなければならないことに注意してください。あるいは、列が、表内の複数の空間列のうちの 1 つである場合は、その列を、その他の空間列をもたないビューに入れることができます。このようにすると、視覚化ツールは、このビューからデータを読み取ることができるようになります。

   シナリオ: 空間管理チームは、空間データを含む列を定義します。

   - LOCATION 列を CUSTOMERS 表に追加します。この表には顧客のアドレスが既に入っています。DB2SE_USA_GEOCODER はそれを空間データに変換します。次に、DB2 はこのデータを LOCATION 列に保管します。
   - 現在、別のファイル・システムに保管されているデータを入れるために、OFFICE_LOCATIONS 表と OFFICE_SALES 表を作成します。このデータには、Safe Harbor 社の支店のアドレス、これらのアドレスからジオコーダーによって導出された空間データ、個々のオフィスから半径 5 マイル以内の領域を定義した空間データがあります。ジオコーダーが導出したデータは、

OFFICE_LOCATIONS 表の LOCATION 列に入り、領域を定義したデータは、OFFICE_SALES 表の SALES_AREA 列に入ります。

5. 必要に応じて、視覚化ツールがアクセスするための空間列をセットアップします。これを行うには、DB2 Spatial Extender カタログに列を登録します。 空間列を登録すると、DB2 Spatial Extender は、その列のすべてのデータは、同一の空間参照系に所属しなければならない、という制約を設定します。 この制約は、データの整合性に必要なもので、ほとんどの視覚化ツールの要件になっています。

   シナリオ: 空間管理チームは、視覚化ツールを使用して、LOCATION 列と SALES_AREA 列の説明を、地図上でグラフィカルに表現する予定です。したがって、チームは、3 つの列をすべて登録します。

6. 空間列にデータを取り込みます。

   空間データをインポートする必要があるプロジェクトでは、データをインポートします。

   ジオコーダーを必要とするプロジェクトでは、次のようにします。

   - ジオコーダーを呼び出すときに必要な制御情報を、前もって設定します。
   - オプションとして、新規アドレスがデータベースに追加されるたび、あるいは既存アドレスが更新されるたびにジオコーダーが自動的に実行されるようにセットアップします。

   必要に応じて、バッチ・モードでジオコーダーを実行します。

   空間処理関数によって空間データを作成する必要があるプロジェクトでは、この関数を実行します。

   シナリオ: 空間管理チームは、CUSTOMER 表の LOCATION 列、OFFICE_LOCATIONS 表、OFFICE_SALES 表、および新しい HAZARD_ZONES 表にデータを取り込みます。

   - チームは、DB2SE_USA_GEOCODER を使用して、CUSTOMER 表のアドレスをジオコーディングします。ジオコーディングによって生成された座標は、表の LOCATION 列に設定されます。
   - チームは、オフィス・データをファイル・システムからファイルにロードするユーティリティーを使用します。次に、チームは、このデータを新規の OFFICE_LOCATIONS 表にインポートします。
   - チームは、HAZARD_ ZONES 表を作成し、その空間列を登録し、その列にデータをインポートします。データは地図の製作会社が提供するファイルにあります。

7. 必要に応じて、空間列へのアクセスを容易にします。それには、DB2 で空間データに素早くアクセスできるようにするための索引を定義し、相互に関係のあるデータを効果的に検索できるようにするためのビューを定義します。 視覚化ツールでビューの空間列にアクセスする場合は、同様にこれらの列を DB2 Spatial Extender に登録する必要があります。

シナリオ: 空間管理チームは、登録された列の索引を作成します。次に、CUSTOMERS 表と HAZARD_ZONES 表の列を結合したビューを作成します。次に、このビューに空間列を登録します。

8. 空間情報と、関連した業務情報を生成し、この情報を分析します。このタスクには、空間列および空間列以外の関連する列の照会が含まれます。この照会に、例えば、危険な廃棄物処理サイトを囲む安全ゾーン案を定義する座標や、そのサイトと近くの公共建物との間の最短距離などの幅広い情報を戻す、DB2 Spatial Extender 関数を組み込むことができます。

   シナリオ: 空間分析チームは、照会を実行して、元の目標 (新しい支店を設立する場所の決定と、顧客と危険地域との近接度に応じた保険料の調整) を果たすのに役立つ情報を取得します。

第 4 章 DB2 Spatial Extender 入門

この章では、Spatial Extender (AIX®、HP-UX、Windows®, Linux®、Linux on IBM System z®、および Solaris オペレーティング環境版) をインストールし、構成する方法を説明しています。またこの章では、Spatial Extender を呼び出した時に起こる可能性のある、インストール上、および構成上のいくつかの問題を解決する方法も説明しています。

Spatial Extender のセットアップとインストール

前提条件

DB2 Spatial Extender をセットアップする前に、DB2 データ・サーバーおよびクライアントを単一のコンピューターにインストールしているか、または、DB2 データ・サーバーを 1 つのコンピューターに、DB2 クライアントを別のコンピューターにインストールしている必要があります。

DB2 Spatial Extender 環境は、DB2 データ・サーバーのインストールと、DB2 Spatial Extender のインストールで構成されます。空間操作用に設定されたデータベースは、DB2 Spatial Extender のクライアントからアクセスできる DB2 データ・サーバー上に置かれます。 DB2 データ・サーバーと DB2 Spatial Extender のクライアントは、同じコンピューターにインストールできます。データベースにある空間データは、DB2 Spatial Extender ストアード・プロシージャーおよび空間照会によってアクセスできます。DB2 Geodetic Data Management Feature は、DB2 Spatial Extender に組み込まれていますが、これを使用するには別のライセンスが必要です。また、通常は、標準的な DB2 Spatial Extender システムの一部であるジオブラウザーは必ずしも必要ではありませんが、空間照会の結果を (一般的には地図の形で) 視覚的に表示させる場合に便利です。ジオブラウザーは DB2 Spatial Extender インストールには組み込まれていませんが、空間データは、オープン・ソースのジオブラウザー、ArcExplorer for DB2、ESRI の ArcSDE と一緒に実行される ArcGIS ツール製品群などのジオブラウザーを使用して表示できます。

このタスクは次のように行います。

1. ご使用のシステムが、すべてのソフトウェア要件に適合していることを確認します。 「*Spatial Extender および Geodetic Data Management Feature ユーザーズ・ガイドおよびリファレンス*」の『Spatial Extender をインストールするためのシステム要件』を参照してください。
2. Spatial Extender をインストールします。システムがソフトウェア前提条件のいずれかを満たさない場合は、インストールは失敗します。 21 ページの『DB2 Spatial Extender のインストール (Windows)』 または 22 ページの『DB2 Spatial Extender のインストール (Linux、UNIX)』 を参照してください。
3. DB2 インスタンスがない場合は、ここで作成します。その場合、DB2 コマンド・ウィンドウから db2icrt DB2 コマンドを使用します。

© Copyright IBM Corp. 1998, 2009

4. Spatial Extender 環境をテストして、Spatial Extender のインストールが正常に実行されたかどうかを検証します。 23 ページの『Spatial Extender のインストールの検査』 を参照してください。
5. オプション: ArcExplorer for DB2 または ESRI の ArcSDE と一緒に実行される ArcGIS ツール製品群などのジオブラウザーをダウンロードしてインストールします。 ArcExplorer for DB2 の無料コピーは、IBM DB2 Spatial Extender の下記の Web サイトからダウンロードすることができます。http://www.ibm.com/software/data/spatial/db2spatial/

Spatial Extender をインストールするためのシステム要件

DB2 Spatial Extender をインストールする前に、以下のすべてのソフトウェアおよびディスク・スペース要件を、システムが満たしていることを確認してください。

オペレーティング・システム

DB2 Spatial Extender は、Windows、または Intel ベースのシステム上の Linux などの 32 ビットのオペレーティング・システムにインストールできます。DB2 Spatial Extender は、AIX、HP-UX PA-RISC、Solaris Operating System SPARC、Linux x86、Linux for System z、および Windows などの、64 ビットのオペレーティング・システムにもインストールできます。

ソフトウェア要件

Spatial Extender をインストールするには、以下の DB2 ソフトウェアをインストールし、構成しておく必要があります。

サーバー・ソフトウェア

DB2 Spatial Extender をインストールする*前* に、DB2 をインストールする必要があります。

DB2 コントロール・センターのグラフィカル・ユーザー・インターフェースを使用して、Spatial Extender タスクのいくつかを実行できます。このインターフェースを使用する可能性がある場合は、DB2 Administration Server (DAS) を作成し、構成します。

コントロール・センターの Spatial Extender サポートについて詳しくは、505 ページの『第 27 章 DB2 コントロール・センターからの空間タスク』を参照してください。

DAS を作成し、構成する方法について詳しくは、「DB2 サーバー機能 インストール」の 『DB2 Administration Server の作成 (Linux および UNIX)』を参照してください。

空間クライアント・ソフトウェア

DB2 Spatial Extender を Windows 上にインストールする場合、 Spatial Extender のデフォルトのインストールには空間クライアントが含まれています。 DB2 Spatial Extender を AIX、HP-UX、Solaris オペレーティング・システム、Linux for Intel®、または Linux on System z オペレーティング・システム上にインストールする場合は、DB2 データ・サーバーおよびクライアントをインストールするときに、オプションで空間クライアントをインストールすることができます。

ディスク・スペース要件

Spatial Extender をインストールするには、システムはインストール処理中に示されるディスク・スペース要件を満たす必要があります。要件が満たされない場合、インストールは失敗し、使用できるディスク・スペースが不十分であるというエラー・メッセージが出ます。

DB2 Spatial Extender のインストール (Windows)

DB2 Spatial Extender を Windows オペレーティング・システムにインストールするには、以下のタスクを正常に完了している必要があります。

DB2 Spatial Extender 機能をインストールする前に、DB2 データ・サーバー製品をインストールしておく必要があります。

このタスクは、 19 ページの『Spatial Extender のセットアップとインストール』というタスクの一部です。

DB2 Spatial Extender は、「DB2 セットアップ」ウィザードまたは応答ファイルを使用することによって、Windows オペレーティング・システム上にインストールできます。

推奨:
「DB2 セットアップ」ウィザードを使用して Spatial Extender をインストールしてください。このセットアップ・ウィザードでは、使いやすいグラフィカル・インターフェースを使用しています。このインターフェースを使用して、インスタンスの作成、ユーザーやグループ作成の自動化、プロトコル構成のセットアップ、およびインストール操作のヘルプへのアクセスなどを行うことができます。

「DB2 セットアップ」ウィザードを使用して Spatial Extender をインストールする場合は、「**キャンセル**」をクリックすれば、インストール作業のどの時点でも、このプロセスを終了できます。

インストールの間いつでも、「ヘルプ」をクリックして、オンライン・インストール操作のヘルプを立ち上げることができます。

セットアップ・ウィザードを使用した Spatial Extender のインストール

1. インストールを実行するために使用するユーザー・アカウントで、システムにログオンします。
2. Spatial Extender の CD または DVD を CD または DVD ドライブに挿入し、マウントします。セットアップ・プログラムが自動的に開かず、setup.exe が自動的に実行しない場合、このコマンドを発行してセットアップ・プログラムを実行します。
3. コマンド・プロンプトから setup コマンドを発行して、セットアップ・プログラムを実行します。
4. インストールが終了した後、示されるログ・ファイルで警告またはエラー・メッセージがないか確認します。

インストールは正常に完了するはずです。インストールのプロセス中にエラーがあった場合は、インストールは停止し、取り消されます。

応答ファイルを使用した Spatial Extender のインストール

「DB2 サーバー機能 インストール」の 『応答ファイルを使用した DB2 製品のインストール (Windows)』を参照してください。

DB2 Spatial Extender のインストール (Linux、UNIX)

DB2 Spatial Extender を Linux または UNIX オペレーティング・システムにインストールするには、以下のタスクを正常に完了している必要があります。

前提条件

DB2 Spatial Extender 機能をインストールする前に、DB2 データ・サーバー製品をインストールしておく必要があります。

このタスクは、 19 ページの『Spatial Extender のセットアップとインストール』というタスクの一部です。

DB2 Spatial Extender は、Linux および UNIX オペレーティング・システムに、「DB2 セットアップ」ウィザード、db2_install コマンド、または応答ファイルを使用してインストールできます。

推奨:「DB2 セットアップ」ウィザードを使用して Spatial Extender をインストールしてください。このセットアップ・ウィザードでは、使いやすいグラフィカル・インターフェースを使用しています。このインターフェースを使用して、インスタンスの作成、ユーザーやグループ作成の自動化、プロトコル構成のセットアップ、およびインストール操作のヘルプへのアクセスなどを行うことができます。

「DB2 セットアップ」ウィザードを使用して Spatial Extender をインストールする場合は、「**キャンセル**」をクリックすれば、インストール作業のどの時点でも、このプロセスを終了できます。

インストールの間いつでも、「ヘルプ」をクリックして、オンライン・インストール操作のヘルプを立ち上げることができます。

Spatial Extender のインストール後に、DB2 インスタンス環境をまだ作成していない場合は作成し、次に、インストールしたものを検証します。

DB2 セットアップ・ウィザードを使用した DB2 Spatial Extender のインストール (Linux、UNIX)

1. Spatial Extender の CD または DVD をクライアント・コンピューターの CD または DVD ドライブに挿入し、マウントします。DB2 Spatial Extender をインストールするときのインターフェースとなる DB2 セットアップ・ランチパッドがオープンします。
2. インストールする製品として DB2 Spatial Extender を選択し、「**次へ**」をクリックします。

3. 「**製品のインストール**」をクリックします。
4. 「**既存の処理**」ボタンをクリックします。 Spatial Extender をインストールする DB2 データ・サーバーの既存コピーを選択します。
5. 「**DB2 セットアップ・ウィザードの起動**」をクリックします。「DB2 セットアップ」ウィザード・ウィンドウが開きます。「DB2 セットアップ」ウィザードを使用して、残りのインストールおよびセットアップのステップを進めます。

インストールは正常に完了するはずです。インストールのプロセス中にエラーがあった場合は、インストールは停止し、取り消されます。

db2_install コマンドを使用した Spatial Extender のインストール (Linux、UNIX)

1. 適切な CD を挿入し、マウントします。
2. ./db2_install コマンドを入力して、db2_install スクリプトを開始します。 db2_install スクリプトは、DB2 製品 CD の root ディレクトリーにあります。 db2_install スクリプトは、製品のキーワードを求めるプロンプトを出します。
3. **GSE** と入力して、DB2 Spatial Extender をインストールします。

応答ファイルを使用した Spatial Extender のインストール

「DB2 サーバー機能 インストール」の 『応答ファイルによる DB2 製品のインストール (Linux および UNIX)』を参照してください。

Spatial Extender のインストールの検査

DB2 Spatial Extender のインストール後に、インストールの検証を行うことをお勧めします。

始める前に

Spatial Extender インストールの検証を行う前に、以下の前提条件を満たしている必要があります。

- Spatial Extender がコンピューターにインストールされていなければならない。
- DB2 データ・サーバーがインストールされているコンピューター上に、DB2 インスタンスが作成されている必要がある。

このタスクについて

このタスクは、Spatial Extender のセットアップと構成タスクの後に実行する必要のあるタスクです。 19 ページの『Spatial Extender のセットアップとインストール』を参照してください。このタスクは、DB2 データベースを作成し、DB2 付属の Spatial Extender サンプル・アプリケーションを実行することから成っています。このサンプル・アプリケーションは、DB2 Spatial Extender 機能の主要なセットが正しく作動していることを検証するために使用できます。このテストにより、DB2 Spatial Extender が正しくインストール、構成されたことが十分に検証されます。

手順

このタスクは次のように行います。

1. Linux および UNIX: DB2 インスタンス所有者の役割に相当するユーザー ID でシステムにログオンします。
2. DB2 データベースを作成します。そうするには、DB2 コマンド・ウィンドウを開き、以下を入力します。

   ```
   db2 create database mydb
   ```

   mydb はデータベース名です。
3. ページ・サイズが 8 KB で、最小で 500 ページのサイズがある SYSTEM TEMPORARY 表スペースがあることを確認します。そのような表スペースがない場合、作成方法の詳細について「データベース: *管理の概念および構成リファレンス*」の『TEMPORARY 表スペースの作成』を参照してください。 これは、*runGSEdemo* プログラムを実行するための要件です。
4. DB2 データベース・アプリケーションのヒープ・サイズを増加して、空間データの大きなスペース所要量をデータベースが収容できるようにします。詳しくは、31 ページの『空間データを収容するデータベースの構成』 を参照してください。
5. *runGSEdemo* プログラムがあるディレクトリーへ移動します。
   - Linux および UNIX オペレーティング・システムへ Spatial Extender をインストールする場合は、次のように入力します。

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

     $HOME は、インスタンス所有者のホーム・ディレクトリーです。
   - Windows オペレーティング・システムに Spatial Extender をインストールする場合は、次のように入力します。

     ```
     cd c:¥Program Files¥IBM¥sqllib¥samples¥extenders¥spatial
     ```

     c:¥Program Files¥IBM¥sqllib は、DB2 Spatial Extender をインストールしたディレクトリーです。
6. インストール検査プログラムを実行します。 DB2 コマンド行で、runGseDemo コマンドを入力します。

   ```
   runGseDemo mydb userID password
   ```

   mydb はデータベース名です。

インストール後の考慮事項

Spatial Extender をインストールした後、以下について考慮します。

- オプション: ArcExplorer for DB2 または ESRI の ArcSDE と一緒に実行される ArcGIS ツール製品群などのジオブラウザーをダウンロードしてインストールします。 ArcExplorer for DB2 の無料コピーは、IBM DB2 Spatial Extender の下記の Web サイトからダウンロードすることができます。http://www.ibm.com/software/data/spatial/db2spatial/

ArcExplorer for DB2 のダウンロード

IBM は、Environmental Systems Research Institute (ESRI) が IBM 用に作成したブラウザーを提供しています。これは、DB2 Spatial Extender データを照会した結果を、中間データ・サーバーなしで、直接、ビジュアルに表示できます。このブラウザーは ArcExplorer for DB2 です。ArcExplorer for DB2 の無料コピーは、以下の IBM Spatial Extender Web サイトからダウンロードできます。

http://www.ibm.com/software/data/spatial/db2spatial/

ArcExplorer for DB2 のインストールと使用方法については、「*Using ArcExplorer*」を参照してください。この資料は、DB2 Spatial Extender の Web サイトで、ArcExplorer for DB2 製品のダウンロードの一部として入手できます。

重要: ArcExplorer for DB2 を、DB2 とは別のディレクトリーにインストールしてください。

第 5 章 DB2 Spatial Extender バージョン 9.7 のアップグレード

DB2 Spatial Extender バージョン 8、バージョン 9.1、またはバージョン 9.5 をインストールしたシステムで、DB2 Spatial Extender をバージョン 9.7 にアップグレードするには、DB2 Spatial Extender バージョン 9.7 をインストールするだけでは不十分です。これらのシステムに対して適切なアップグレード作業を行う必要があります。

以下の作業では、DB2 Spatial Extender をバージョン 8、バージョン 9.1、またはバージョン 9.5 からバージョン 9.7 にアップグレードするためのすべての手順を説明します。

- 『DB2 Spatial Extender のアップグレード』
- 28 ページの『32 ビット DB2 Spatial Extender から 64 ビット版への更新』

ご使用の DB2 環境に、DB2 サーバー、クライアント、およびデータベース・アプリケーションなどの他のコンポーネントがある場合、これらのコンポーネントをアップグレードする方法の詳細については、「*DB2 バージョン 9.7 へのアップグレード*」の『DB2 バージョン 9.7 へのアップグレード』を参照してください。

DB2 Spatial Extender のアップグレード

DB2 Spatial Extender をアップグレードするには、DB2 サーバーを初めにアップグレードし、次に空間操作が可能なデータベースの特定のデータベース・オブジェクトおよびデータをアップグレードする必要があります。

始める前に

アップグレード処理を開始する前に以下を行ってください。

- 移行プロセスを開始する前に、ご使用のシステムが、DB2 Spatial Extender Version 9.7 のインストール要件を満たしていることを確認する。
- 空間操作可能なデータベースに関する DBADM 権限または DATAACCESS 権限があることを確認する。
- ページ・サイズが 8 KB で、最小で 500 ページのサイズがある SYSTEM TEMPORARY 表スペースがあることを確認します。そのような表スペースがない場合、作成方法の詳細について「*データベース: 管理の概念および構成リファレンス*」の『TEMPORARY 表スペースの作成』を参照してください。

このタスクについて

DB2 Spatial Extender のバージョン 8、バージョン9.1、またはバージョン 9.5 をインストールしてある場合は、空間操作が可能になっている既存のデータベースを DB2 Spatial Extender のバージョン 9.7 または DB2 Geodetic Data Management Feature のバージョン 9.7 で使用する前に、次のステップを完了しておく必要があり

© Copyright IBM Corp. 1998, 2009

ます。ここでは、空間操作が可能になっているデータベースを、直前バージョンの DB2 Spatial Extender からアップグレードする際に必要なステップを説明します。

手順

DB2 Spatial Extender をバージョン 9.7 にアップグレードするには、次のとおりにします。

1. 以下のいずれかのタスクを実行して、DB2 サーバーをバージョン 8、バージョン9.1、または バージョン 9.5 からバージョン 9.7 にアップグレードします。
   - 「*DB2 バージョン 9.7 へのアップグレード*」の『DB2 サーバーのアップグレード (Windows)』
   - 「*DB2 バージョン 9.7 へのアップグレード*」の『DB2 サーバーのアップグレード (Linux および UNIX)』

   インスタンスを正常にマイグレーションするには、アップグレード・タスクにおいて、DB2 バージョン 9.7 をインストールした後に DB2 Spatial Extender バージョン 9.7 をインストールする必要があります。
2. データベースに対するすべての接続を終了します。
3. db2se upgrade コマンドを使用して、空間操作が可能なデータベースをバージョン 8、バージョン 9.1、またはバージョン 9.5 からバージョン 9.7 にアップグレードします。

受け取るエラーの詳細については、メッセージ・ファイルを確認してください。メッセージ・ファイルには、索引、ビュー、およびアップグレードされたジオコーディングのセットアップなどについての有用な情報も含まれています。

32 ビット DB2 Spatial Extender から 64 ビット版への更新

32 ビット DB2 Spatial Extender バージョン 9.7 から 64 ビット DB2 Spatial Extender バージョン 9.7 への更新では、空間インデックスをバックアップし、DB2 サーバーを 64 ビット DB2 バージョン 9.7 にアップデートし、64 ビット DB2 Spatial Extender バージョン 9.7 をインストールした後、バックアップした空間インデックスをリストアすることが必要です。

始める前に

- ページ・サイズが 8 KB で、最小で 500 ページのサイズがある SYSTEM TEMPORARY 表スペースがあることを確認します。そのような表スペースがない場合、作成方法の詳細について「*データベース: 管理の概念および構成リファレンス*」の『TEMPORARY 表スペースの作成』を参照してください。

このタスクについて

32 ビット DB2 Spatial Extender バージョン 8、バージョン 9.1、またはバージョン 9.5 から 64 ビット DB2 Spatial Extender バージョン 9.7、にアップグレードしている場合は、代わりに 27 ページの『DB2 Spatial Extender のアップグレード』 タスクを実行する必要があります。

手順

32 ビット DB2 Spatial Extender バージョン 9.7 サーバーを 64 ビット DB2 Spatial Extender バージョン 9.7 にアップグレードするには、次のとおりにします。

1. データベースをバックアップします。
2. オペレーティング・システムのコマンド・プロンプトから db2se save_indexes コマンドを発行して、定義された空間インデックスを保管します。
3. 以下のいずれかのタスクを実行して、32 ビット DB2 サーバーをバージョン 8、バージョン 9.1、またはバージョン 9.5 から 64 ビット DB2 バージョン9.7 にアップグレードします。
   - 「*DB2 サーバー機能 インストール* 」の『32 ビット DB2 インスタンスから 64 ビット・インスタンスへの更新 (Windows)』
   - 「*データベース: 管理の概念および構成リファレンス*」の『DB2 コピーの更新 (Linux および UNIX)』
4. DB2 Spatial Extender バージョン 9.7 をインストールします。
5. オペレーティング・システムのプロンプトから db2se restore_indexes コマンドを発行して空間インデックスをリストアします。

第 6 章 データベースのセットアップ

この章では、空間データを入れるデータベースを構成する方法を説明しています。

空間データを収容するデータベースの構成

このトピックでは、DB2 Spatial Extender の操作に影響を与える DB2 構成パラメーターを示します。

DB2 データベース環境で実行される DB2 Spatial Extender は、ほとんどのデフォルト DB2 構成値で動作します。ただし、空間操作に影響を与える構成パラメーターがいくつかあります。空間アプリケーションをできるだけ効率よく実行できるように、これらのパラメーターを調整する必要があります。あるデータベースに対してこれらのパラメーター値を変更すると、変更内容はそのデータベースにだけ影響します。空間操作のために、デフォルト値以外の値を選択しなければならない場合もあります。また、アプリケーションや DB2 全体の環境によっては、そうすることをお勧めする場合もあります。

次のセクションでは、DB2 Spatial Extender の操作に影響を与える、 DB2 データベース・マネージャー・パラメーターとデータベース構成パラメーターの調整方法について説明します。

トランザクション・ログ特性のチューニング

データベースを空間操作に使用できるようにする前に、十分なトランザクション・ログ容量を確保するようにしてください。以下のような場合には、トランザクション・ログ構成パラメーターのデフォルト値では、トランザクション・ログ容量は十分ではありません。

- Windows 環境でデータベースを空間操作に使用できるようにする
- ST_import_shape ストアード・プロシージャーを使用して、形状ファイルからインポートする
- 大きなコミット効力範囲でジオコーディングする
- 並行してトランザクションを処理する

現在、または将来、以上のどれかの使い方をする場合は、1 つ以上のトランザクション・ログ構成パラメーターの値を大きくして、データベースのトランザクション・ログ容量を増やす必要があります。そのような使い方をしない場合は、デフォルトの特性を使用できます。

推奨: 3 つのトランザクション・ログ構成パラメーターの最小推奨値については、次の表を参照してください。

© Copyright IBM Corp. 1998, 2009

表 1. トランザクション構成パラメーターの最小推奨値

パラメーター	説明	デフォルト値	最小推奨値
LOGFILSIZ	ログ・ファイル・サイズを 4 KB ブロックの数で指定する	1000	1000
LOGPRIMARY	1 次ログ・ファイルの内、何個を事前にリカバリー・ログ・ファイルに割り振るかを指定する	3	10
LOGSECOND	2 次ログ・ファイルの数を指定する	2	2

トランザクション・ログの容量が適切でない場合、データベースを空間操作に使用可能にしようとすると、次のエラー・メッセージが出されます。

```
GSE0010N DB2 に十分なログ・スペースが有りません。
```

1 つ以上の構成パラメーターの値を増やすには、次のようにします。

1. コマンド GET DATABASE CONFIGURATION を発行して、LOGFILSIZ、LOGPRIMARY、および LOGSECOND パラメーターの現行値を確認するか、または DB2 コントロール・センターの「**データベースの構成**」ウィンドウを表示します。
2. 上記の表に示された値のうち、変更する個数を 1 つ、2 つ、または 3 つのどれにするかを決めます。
3. 変更しようとする各値を変えます。 次のコマンドを 1 つ以上出すことによって、値を変更できます。*db_name* は使用するデータベースを示します。

```
UPDATE DATABASE CONFIGURATION FOR db_name USING LOGFILSIZ 1000

UPDATE DATABASE CONFIGURATION FOR db_name USING LOGPRIMARY 10

UPDATE DATABASE CONFIGURATION FOR db_name USING LOGSECOND 2
```

LOGSECOND パラメーターだけを変更した場合は、変更内容は即時に有効になります。

LOGFILSIZ パラメーターか LOGPRIMARY パラメーター、またはその両方を変更した場合は、次のようにします。

1. すべてのアプリケーションをデータベースから切断します。
2. データベースが明示的にアクティブ化されている場合は、データベースを非活動化します。

LOGFILSIZ パラメーターや LOGPRIMARY パラメーター、またはその両方に対する変更は、次にデータベースがアクティブ化されるか、あるいは次にデータベースへの接続が確立されるときに有効になります。

アプリケーション・ヒープ・サイズのチューニング

アプリケーション・ヒープ・サイズ (4 KB ページの数) を指定するには、データベース構成パラメーター APPLHEAPSZ を使用します。このパラメーターは、特定のエージェントやサブエージェントのために、データベース・マネージャーが使用できる専用メモリーのページ数を定義します。ヒープは、エージェントやサブエージェントがアプリケーションに対して初期設定されるときに割り振られます。割り振られる量は、エージェントやサブエージェントに対する要求を処理するのに必要な最小量です。より大きな SQL ステートメントを処理するために、エージェントやサブエージェントがより多くのヒープ・スペースを必要とする場合は、データベース・マネージャーが必要に応じてメモリーを割り当てます。最大量は、このパラメーターで指定されます。アプリケーション・ヒープは、エージェントの専用メモリーから割り振られます。

APPLHEAPSZ パラメーターのデフォルト値は 128 (4 KB ページの数) です。ST_enable_db ストアード・プロシージャーを実行する場合は、この値は 2048 以上であることが必要です。

推奨: ほとんどの DB2 Spatial Extender アプリケーションの場合、特に形状ファイルとの間でインポートやエクスポートを行うアプリケーションでは、2048 以上の APPLHEAPSZ パラメーター値を使用してください。

APPLHEAPSZ の値が適切でないと、データベースを空間操作に使用可能にしようとすると、次のエラー・メッセージが出されます。

```
GSE0009N DB2 アプリケーション・ヒープに十分なスペースが有りません。

GSE0213N バインド操作が失敗しました。SQLERROR = "SQL0001N バインド、または
プリコンパイルが正常に完了しませんでした。SQLSTATE=00000".
```

アプリケーション・ヒープ・サイズを変更するには、次のようにします。

1. コマンド GET DATABASE CONFIGURATION を発行して APPLHEAPSZ パラメーターの現行値を確認するか、または DB2 コントロール・センターの「**データベースの構成**」ウィンドウを表示します。
2. 値を推奨値の 2048 以上に変更します。 次のコマンドを出すことによって、値を 2048 に変更できます。*db_name* は使用するデータベースを示します。

   ```
   UPDATE DATABASE CONFIGURATION FOR db_name USING APPLHEAPSZ 2048
   ```
3. すべてのアプリケーションをデータベースから切断します。
4. データベースが明示的にアクティブ化されている場合は、データベースを非活動化します。

変更説明は、次にデータベースがアクティブ化されるか、あるいは、次にデータベースへの接続が確立されるときに有効になります。

第 7 章 データベース用の空間リソースのセットアップ

空間データを入れるデータベースをセットアップすれば、データベースにリソースを提供する準備ができたことになります。これらのリソースは、空間列を作成し、管理し、空間データを分析するために必要なものです。リソースには以下のものがあります。

- 空間の操作をサポートするために Spatial Extender が提供するオブジェクト : 例えば、データベースを管理するストアード・プロシージャー、空間データ、および、空間データのジオコーディングや、インポート、エクスポートを行う空間ユーティリティーなど。
- 参照データ: 個々のアドレスを座標に変換するために DB2SE_USA_GEOCODER が使用する、アドレスの範囲。
- ユーザーまたはベンダーが提供する、任意のジオコーダー。

この章では、これらのリソースを説明し、リソースを使用できるようにするためのタスクを紹介します。そのタスクでは、データベースを空間操作に使用できるようにし、参照データへアクセスをセットアップし、デフォルト以外のジオコーダーの登録を行います。

データベースにリソースをセットアップする方法

データベースに空間データを収容できるようにセットアップした後で実行する最初のタスクは、データベースが空間操作をサポートできるようにすることです。空間操作とは、表に空間データを入れたり、空間の照会を処理したりする操作です。このタスクでは、DB2 Spatial Extender が提供する特定のリソースをデータベースにロードします。このセクションでは、これらのリソースと、タスクの大筋を説明します。

データベースに提供されるリソースのインベントリー

データベースに対する空間操作をサポートするために、DB2® Spatial Extender は、以下のリソースをデータベースに提供しています。

- ストアード・プロシージャー。例えば空間データをインポートするコマンドを出すなどして、空間操作を要求する場合には、DB2 Spatial Extender は、その操作を実行するために、ストアード・プロシージャーのどれか 1 つを呼び出します。
- 空間データ・タイプ。空間データの格納先となる個々の表またはビューの列には、空間データを割り当てなければなりません。
- DB2 Spatial Extender カタログ。 このカタログに従属している操作もあります。例えば、視覚化ツールから空間列にアクセスする場合、ツールによっては、前もって、その空間列をカタログに登録しておく必要があります。
- 空間グリッド索引。空間列にグリッド索引を定義できます。
- 空間処理関数。これらを使用して、さまざまな方法で空間データを処理できます。例えば、形状間のリレーションシップを判別したり、空間データをさらに生成したりできます。

© Copyright IBM Corp. 1998, 2009

- 座標システムの定義。
- デフォルトの空間参照系。
- 2 つのスキーマ: DB2GSE および ST_INFORMTN_SCHEMA。DB2GSE には、ストアード・プロシージャー、空間データ、DB2 Spatial Extender カタログなどのオブジェクトが含まれています。 カタログのビューは、ST_INFORMTN_SCHEMA でも使用できます。

データベースに対する空間操作の使用可能化

始める前に

空間操作にデータベースを使用できるようにする前に、次のことを行ってください。

- 使用しているユーザー ID がデータベースに対する DBADM 権限を持っていることを確認します。
- ページ・サイズが 8 KB で、最小で 500 ページのサイズがある SYSTEM TEMPORARY 表スペースがあることを確認します。そのような表スペースがない場合、作成方法の詳細について「*データベース: 管理の概念および構成リファレンス*」の『TEMPORARY 表スペースの作成』を参照してください。

このタスクについて

DB2 Spatial Extender によって、空間列を作成したり空間データを操作するためのリソースをデータベースに提供するようにするタスクのことを、通常、『データベースに対する空間操作の使用可能化』と呼びます。

手順

以下のどれかの方法によって、データベースに対する空間操作を使用可能にできます。

- DB2 Spatial Extender のメニュー・オプションから「データベースの使用可能化 (Enable Database)」ウィンドウを使用する。メニュー・オプションは、DB2 コントロール・センターのデータベース・オブジェクトから使用できます。
- db2se enable_db コマンドを発行する。
- db2gse.ST_enable_db ストアード・プロシージャーを呼び出すアプリケーションを実行する。

DB2 Spatial Extender カタログを常駐させる表スペースを、明示的に選択することができます。選択しないと、DB2 はデフォルトの表スペースを使用します。

参照データでの作業

このセクションでは、参照データとは何か、およびこれにアクセスするには何をする必要があるかを説明しています。

参照データ

参照データとは、個々のアドレスを座標に変換するために DB2SE_USA_GEOCODER が使用する、アドレスの集まりのことです。このデータは、米国の国勢調査局が集計した、最新のアドレスで構成されています。DB2SE_USA_GEOCODER がデータベースからアドレスを読み取る時には、次のものを参照データから検索します。

- アドレスの郵便番号によって指定された領域内にある特定の道路名。ジオコーダーは、アドレスにある道路名に、指定された度合い、または指定された度合いより高い度合い (例えば 80 % またはそれ以上) で一致する名前を探します。
- アドレス番号に対応するアドレスの範囲。

一致したものが見付かり、要求されたスコアがなかった場合は、ジオコーダーは、読み取ったアドレスの座標を戻します。一致したものが見つからないか、または要求されたスコアがない場合は、ジオコーダーは NULL を戻します。

ロケーター・ファイル と呼ばれる拡張構成ファイルを使用して、ジオコーダー DB2SE_USA_GEOCODER による処理を操作することができます。DB2® Spatial Extender が提供するデフォルトの構成は、通常は、このファイルで変更する必要はありません。

DB2SE_USA_GEOCODER の参照データに対するアクセスのセットアップ

DB2SE_USA_GEOCODER の参照データをダウンロードできます。ここでは、ジオコーダー参照データにアクセスするための準備について説明します。

始める前に

ジオコーダー参照データを入れるのに十分なスペース (約 700 MB) があることを確認する。

手順

DB2SE_USA_GEOCODER の参照データに対するアクセスをセットアップするには、次のようにします。

1. http://www.ibm.com/software/data/spatial/db2spatial にあるトライアルとデモの Web ページで DB2 Spatial Extender Sample Map Data オファリングを選択することにより、ジオコーダー参照データの zip ファイルをダウンロードし、その zip ファイルを以下のいずれかのディレクトリーに解凍します。
   - $**DB2INSTANCE**/sqllib/gse/refdata/ (Linux または UNIX オペレーティング・システムの場合)
   - %**DB2PATH**%¥gse¥refdata¥ (Windows オペレーティング・システムの場合)

   内容については、参照データに付いている README ファイルを参照してください。
2. DB2SE_USA_GEOCODER に、ロケーター・ファイルと基本地図の名前とロケーションを知らせる。これを行うには、DB2SE_USA_GEOCODER の base_map と

locator_file のパラメーターを適切な値に設定します。詳細については、データベース管理者に確認するか、IBM 担当員に連絡してください。

ジオコーダーの登録

データベースを空間操作に使用できるようにすると、DB2SE_USA_GEOCODER は自動的に DB2 Spatial Extender に登録されます。他のジオコーダーを使用するには、そのジオコーダーを登録する必要があります。

始める前に

ジオコーダーを登録するには、ジオコーダーがあるデータベースに対して、ユーザー ID が DBADM 権限をもっている必要があります。

手順

以下のどれかの方法によって、ジオコーダーを登録できます。

- DB2 コントロール・センターの「ジオコーダーの登録」ウィンドウから登録する。
- db2se register_gc コマンドを発行する。
- db2gse.ST_register_geocoder ストアード・プロシージャーを呼び出すアプリケーションを実行する。

第 8 章 プロジェクト用の空間リソースのセットアップ

データベースを空間操作に使用できるようにすると、空間データを使用するプロジェクトを作成できるようになります。各プロジェクトが必要とするリソースの中には、空間データが準拠する座標システム、および、データが参照する地理上の地域の範囲を定義する、空間参照系があります。この章では、以下の項目を説明しています。

- 座標システムの性質とその作成方法を説明します。
- 空間参照系とは何か、およびその作成方法を説明します。

座標システムの使用法

空間データを使用するプロジェクトを計画する場合、そのデータが、Spatial Extender カタログに登録されている座標システムの 1 つに基づくべきかどうかを判断する必要があります。登録されている座標システムに、要件を満たすものがない場合は、要件を満たす座標システムを作成します。ここでは、座標システムの概念を説明し、座標システムを選択して使用する方法、および新しい座標システムを作成する方法を紹介します。

座標システム

座標システムは、特定領域にあるものの相対的な位置を定義するためのフレームワークです。例えば、地球表面のある領域や地球表面全体を定義できます。DB2® Spatial Extender では、地形の位置を確定するための以下のタイプの座標系がサポートされています。

地理座標システム
*地理座標*システムは、地球上の位置を確定するために 3 ディメンションの球体の表面を利用する参照系です。地球上のすべての位置は、角度を測定単位とする緯度座標、経度座標を持ったポイントとして表現できます。

投影座標システム
*投影座標*システム は、地球を、平面的な 2 ディメンションで表現したものです。線形測定単位を基礎とする直交 (Cartesian) 座標が利用されます。この座標システムは、球体 (回転楕円体) 地球モデルを基礎としており、その座標は、投影トランスフォーメーションによって地理座標に関連付けられます。

地理座標システム

*地理座標*システム は、地球上の位置を確定するために 3 ディメンションの球体の表面を利用します。地球上のすべての位置は、緯度と経度による点で表すことができます。その点の値には、次の測定単位が使用できます。

- 地理座標システムが DB2® Geodetic Data Management Feature の認識する SRID (Spatial Reference System Identifier) を持つ場合の線形単位。

© Copyright IBM Corp. 1998, 2009

- 地理座標システムが DB2 Geodetic Data Management Feature の認識しない SRID を持つ場合は、以下のうちのいずれか。
  - 度 (10進数)
  - 分 (10進数)
  - 秒 (10進数)
  - グラジアン
  - ラジアン

たとえば、図 6 は、地理座標システムにおける、東経 80 度、北緯 55 度の地点を示しています。

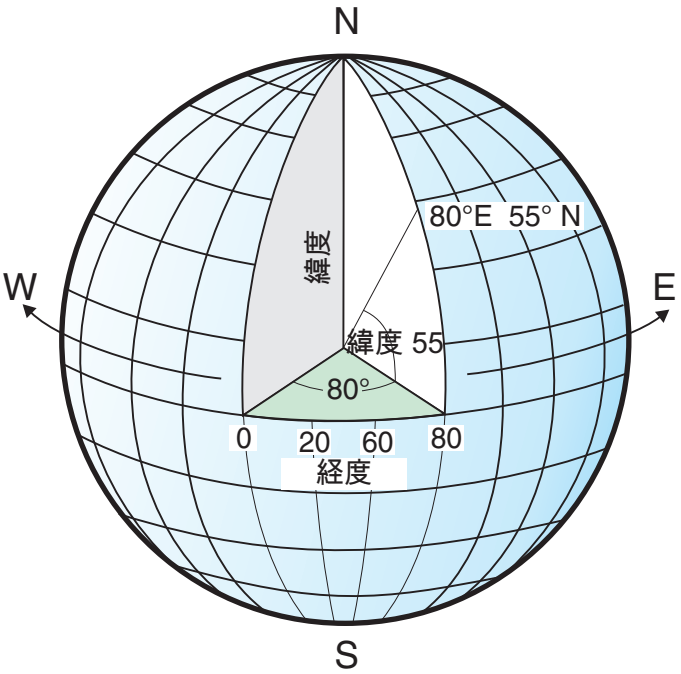


図 6. 地理座標システム

地球上を東西に走る線は*緯線* と呼ばれます。同じ緯線上の地点はどれも同じ緯度を持ちます。水平線は、互いに平行で、等距離間隔にあり、地球の周りに同心円を形成します。*赤道* は最長の緯線であり、これにより、地球は 2 分されます。各極からの赤道までの距離は等しく、この線の値はゼロ度です。赤道より北の地点は、0 度から +90 度までの正の緯度をもち、赤道より南の地点は、0 度から -90 度までの負の緯度をもちます。

41 ページの図 7 は緯線を表しています。

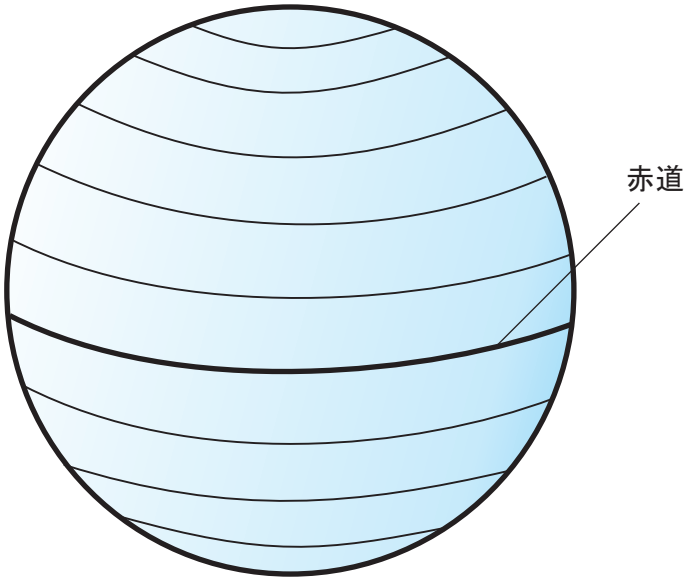


図 7. 緯線

地球上を南北に走る線は*子午線 (経線)* と呼ばれます。同じ経線上の地点はどれも同じ経度を持ちます。経線は、地球の周りに同じサイズの円を形成し、極で交差します。*本初子午線* は、経度座標の起点 (ゼロ度) を定義する経線です。最もよく使用される本初子午線の 1 つは、イングランドのグリニッジを通過する子午線です。ただし、他にも本初子午線として使用できる、ベルン、ボゴタ、パリを通過する経線があります。本初子午線より東の、*反対側*の子午線（本初子午線の地球の裏側での延長線）までの地点は、0 度から +180 度までの正の経度をもちます。本初子午線より西の地点は、0 度から –180 度までの負の経度をもちます。

図 8 は経線を表しています。

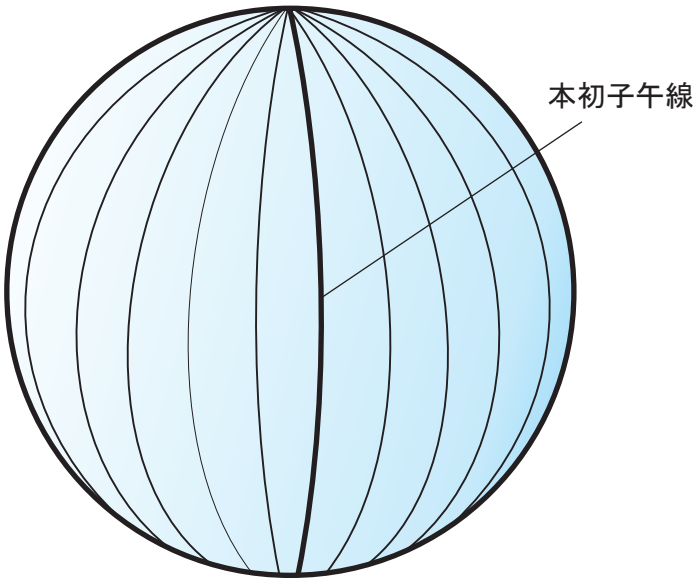


図 8. 経線

経線と緯線は、地球を取り囲む*経緯網* と呼ばれるグリッドの網目を形成します。経緯網の起点は、赤道と本初子午線 (グリニッジ子午線) が交差する場所にあり、(0,0) で表されます。赤道は、緯度 1 度分の直線距離が経度 1 度分の直線距離とほぼ等しい経緯網上の唯一の場所です。経線は極で 1 点に収束するため、各子午線間の距離は緯度によりそれぞれ異なります。したがって、極に近づけば近づくほど、経度 1 度分の距離は緯度 1 度分の距離よりも小さくなります。

また、経緯網を使用して、緯線の長さを判別することも難しい作業です。緯線は同心円であり、極に近づくほど小さくなります。緯線は、子午線が始まる極で単一の点になります。赤道では、経度 1 度分の距離は約 111.321 km ですが、緯度 60 度では、経度 1 度分の距離は 55.802 km になります (1866 年のクラーク楕円体を基にした概算)。このように、緯度と経度に関する統一された長さは存在しないため、角度の測定単位を使用して、ポイント間の距離を正確に測定することは不可能です。

図 9 は、経緯網上では、ロケーションによって長さが異なることを示しています。

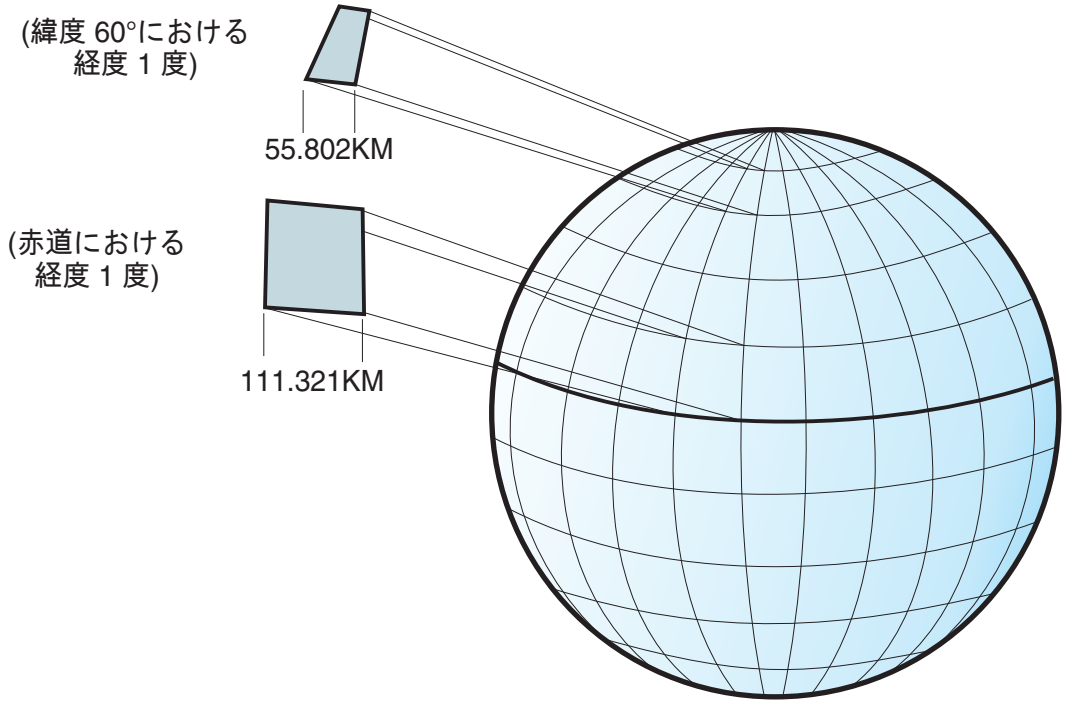


図 9. 経緯網上のロケーションによる長さの違い

座標システムは、地球の形状に似せた球面や回転楕円面によって定義することができます。地球は完全な球体ではないため、表すべき地点によっては、回転楕円面を使用すると、地図の正確に表すのに役立ちます。*回転楕円面* は、楕円に基づいた楕円形物体ですが、球面は円に基づいています。

楕円の形状は、2 つの半径によって決まります。長い半径は半長軸と呼ばれ、短い半径は半短軸と呼ばれます。楕円面は、楕円を軸の 1 つを中心にして回転させることによって作られる 3 ディメンション形状です。

図 10 は、地球に似せた球面と回転楕円体、そして楕円の長軸と短軸を示したものです。

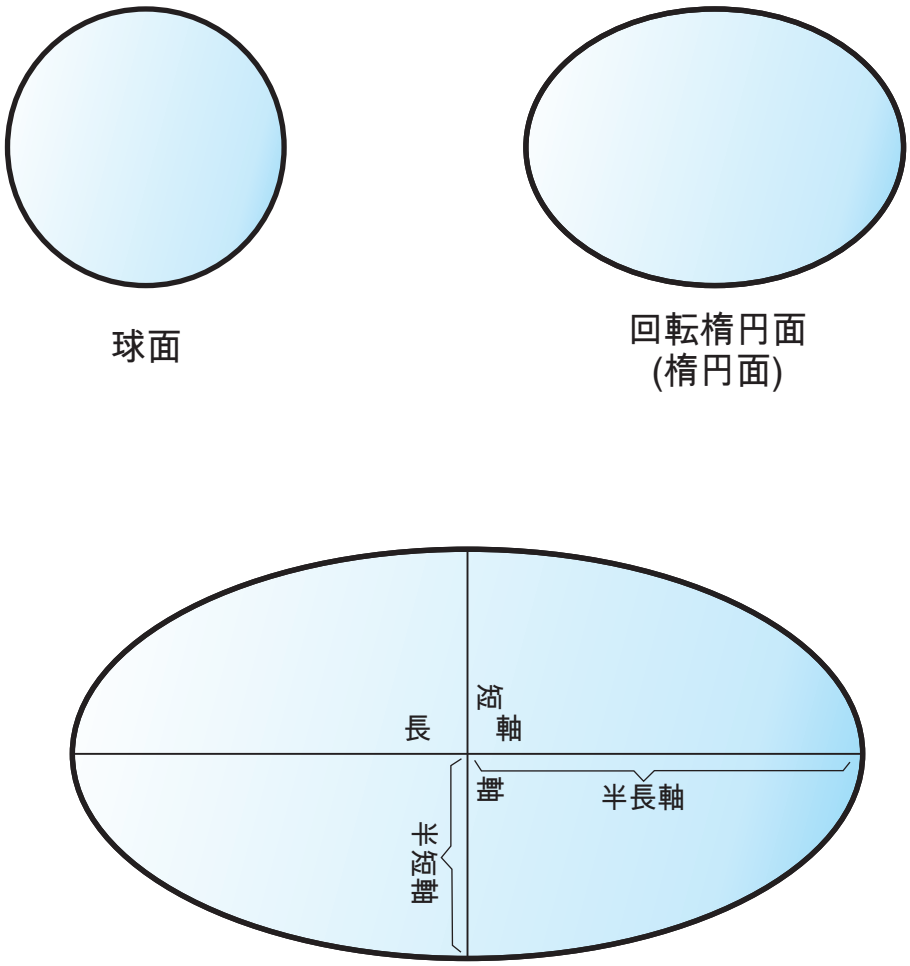


図 10. 球面と回転楕円面の近似化

測地系 は、地球の中心に対する回転楕円面の相対的な位置を定義する値のセットです。測地系は、ロケーションを測定するための参照フレームであり、緯線と経線の起点と方位を定義します。測地系の中には、世界中のどの地点でも一定の正確性を保つことを目的とした「グローバル」の測地系もあります。 ローカルの測地系は、特定領域の地球表面に厳密に適合するように、回転楕円面を並べます。したがって、座標システムの測定は、それらが使用される予定以外の領域に対して使用されると、正確でなくなります。

44 ページの図 11 は、地球表面に対する異なる測地系の配置を示します。ローカルの測地系 NAD27 は、地球中心の測地系 WGS84 に比べ、特定の地点に関しては実際の地球表面に近いものになります。

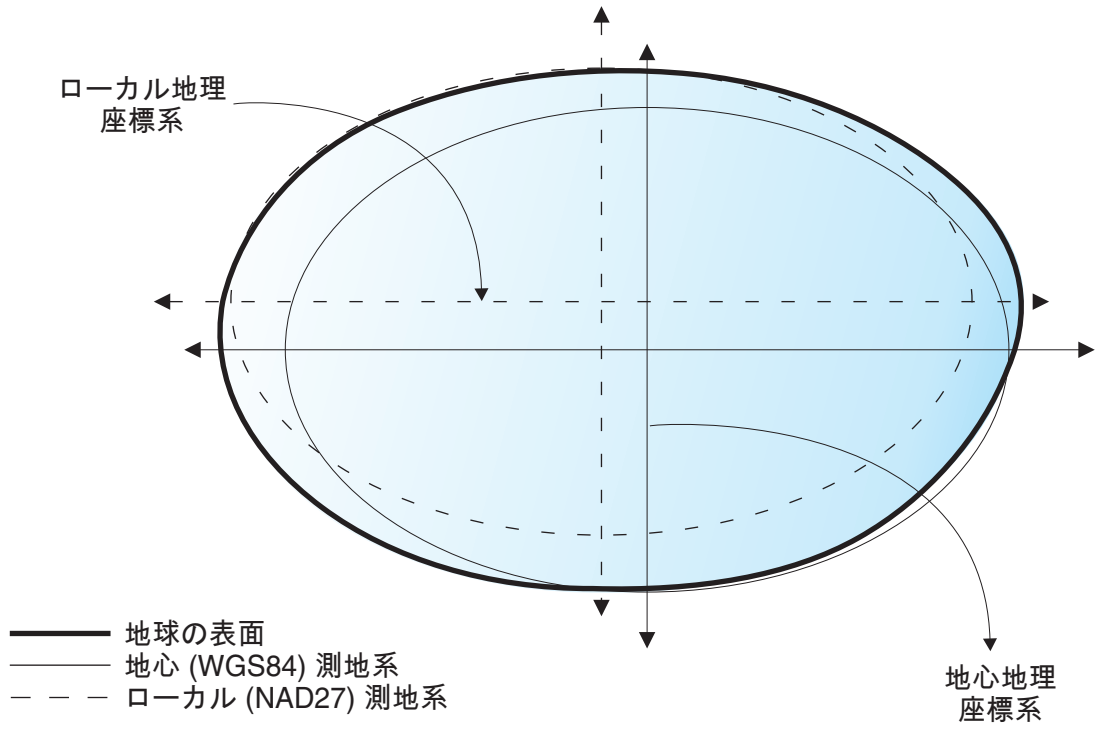


図 11. 測地系の配置

測地系を変更すると、必ず、地理座標システムが変更され、座標の値も変更されます。たとえば、North American Datum of 1983 (NAD 1983) を使用したカリフォルニア州 Redlands のコントロール・ポイントの DMS による座標は「-117 12 57.75961 34 01 43.77884」ですが、North American Datum of 1927 (NAD 1927) を使用した同じポイントの座標は「-117 12 54.61539 34 01 43.72995」です。

投影座標システム

投影座標システム は、地球を、平面的な 2 ディメンションで表現したものです。投影座標システムは、球面や回転楕円面の地理座標システムに基づいていますが、座標については、線形の測定単位を使用するため、距離や面積の計算は、同様の単位を使用して簡単に行うことができます。

緯度と経度の座標は、投影平面上の X と Y の座標に変換されます。x 座標は、通常、あるポイントの東方向を、y 座標はあるポイントの北方向を示します。中央を東から西に走る線は、x 軸、北から南に走る線は y 軸と呼ばれます。

x 軸と y 軸の交点が原点で、通常、座標は (0,0) である。X 軸より上の値は正の値であり、X 軸より下の値は負の値です。x 軸に平行な線は等間隔で引かれます。Y 軸より右の値は正の値であり、Y 軸より左の値は負の値です。y 軸に平行な線は等間隔で引かれます。

3 ディメンションの地理座標システムを 2 ディメンションの平面の投影座標システムに変換するためには、数式が使用されます。この変換は*地図投影* と呼ばれます。通常、地図投影は、円すい、円柱、平面の表面などの投影表面によって分類されます。使用される投影によって、異なる空間プロパティーは、ゆがめられて表示されます。投影は、1 つまたは 2 つのデータ特性のゆがみを最小限にするように設計さ

れていますが、距離、面積、形状、方向、またはこれらのプロパティーの組み合わせは、モデル化されているデータを正確に表現しない場合があります。投影のタイプにはいくつかあります。ほとんどの地図投影は、空間プロパティーをある程度正確に保存しようとしますが、この方法とは異なり、*Robinson* 投影などのように、全体のゆがみを最小限にしようとするものもあります。地図投影の最も一般的なタイプには、以下のものがあります。

正積図法

この図法では特定の地形の面積を保持します。これらの射影は、形状、角度、距離をゆがめます。正積図法の例として、*アルベルス正積円すい図法 (Albers Equal Area Conic)* があります。

正角図法

この投影は、小さな領域のローカルな形状を保持します。地図上で 90 度の角度で交差する直角の経緯網を表示することによって、空間リレーションシップを記述する個々の角度を保持します。すべての角度は保持されますが、地図の面積はゆがめられます。正角図法の例には、*メルカトル (Mercator)* とランベルト等角円すい図法 *(Lambert Conformal Conic)* があります。

正距図法

この投影では、特定のデータ・セットの縮尺を維持することによって、特定のポイントの間の距離を保持します。距離によっては、実際の距離に一致するものもあります。実際の距離とは、地球と同じ尺度で同じ距離ということです。このデータ・セット以外の部分では、距離はよりゆがめられるようになります。正距図法の例としては、*正弦曲線 (Sinusoidal)* 図法と*正距 (Equidistant Conic)* 図法があります。

方位図法

この投影では、大圏の一部を維持することによって、あるポイントからその他のすべてのポイントに対する方向を保持します。この図法は、地図上のすべてのポイントの、中心を基準とした方向または方位角を正確に与えます。方位角地図は、正積図法、正角図法、正距図法と組み合わせることができます。方位図法の例として、ランベルト正積方位 *(Lambert Equal Area Azimuthal)* 図法と*正距方位 (Azimuthal Equidistant)* 図法があります。

座標システムを選択または作成する

プロジェクトを計画するには、まず、どの座標システムを使用するかを決めます。

座標システムを作成するには、その前に、ユーザー ID が、空間操作に対して使用可能にされたデータベースに関して、DBADM 権限をもっていなければなりません。既存の座標システムを使用する場合には、権限は必要ありません。

データベースを空間操作に使用できるようにすると、空間データを使用するプロジェクトを計画できるようになります。DB2 Spatial Extender に付いてくる座標システム、または他の場所で作成された座標システムを使用できます。DB2 Spatial Extender では、2000 を超える座標システムを使用できます。以下のものがあります。

- DB2 Spatial Extender で、Unspecified と呼ばれる座標システム。以下の場合に、この座標システムを使用します。

  - 地球表面と直接のリレーションシップをもっていないロケーションを定義する必要がある。例えば、オフィスビル内のオフィスのロケーションまたは収納室内の棚のロケーションなど。
  - 小数値を少し含むか、あるいはまったく含まない、正の座標によってこれらのロケーションを定義できる。
- GCS_NORTH_AMERICAN_1983. 米国のロケーションを定義する必要があるときに、この座標システムを使用します。例えば、以下のような場合です。
  - ダウンロード可能な Spatial Map Data から米国の空間データをインポートする場合。
  - 米国内の住所をジオコーディングするために、DB2 Spatial Extender に付属のジオコーダーを使用する計画がある場合。

これらの座標システムの詳細を参照したり、DB2 Spatial Extender に付いてくるその他の座標システムの内容や他のユーザーが作成した座標システムがあるかどうかを確認するには、DB2SE.ST_COORDINATE_SYSTEMS カタログ・ビューを参照してください。

このタスクは次のように行います。

座標システムの作成に使用する方法を選択します。

- DB2 コントロール・センターの「座標システムの作成」ウィンドウから作成する。
- db2se コマンド行プロセッサーから、db2se create_cs コマンドを発行する。
- db2se.ST_create_coordsys ストアード・プロシージャーを呼び出すアプリケーションを実行する。

空間参照系のセットアップ方法

空間データを使用するプロジェクトを計画する場合、使用可能な空間参照系の中に、このデータに使用できるものがあるかどうかを判断する必要があります。使用できるシステムの中に、そのデータに適切なものがない場合は、新しく作成することができます。このセクションでは、空間参照系の概念および、使用するものを選択するタスクおよび、新しく作成するタスクについて説明します。

空間参照系

空間参照系 は、パラメーターのセットであり、以下のものが含まれます。

- 座標の導出元となっている座標システムの名前。
- 空間参照系を他と区別するための固有の数値 ID。
- 与えられた座標の範囲により参照される、可能な最大範囲のスペースを定義する座標。
- 特定の数学演算で使用した場合、入力として受け取られた座標を、最も効率良く処理できる値に変換する数。

以下のセクションでは、ID 、スペースの最大範囲、変換係数などを定義するパラメーター値について説明します。

空間参照系の ID

空間参照系 ID (SRID) は、さまざまな空間処理関数の入力パラメーターとして使用されます。

測地参照系の場合、SRID 値の範囲は 2000000000 から 2000001000 までででなければなりません。DB2® Geodetic Data Management Feature は、318 の定義済み測地参照系 (SRS) を提供しています。

空間列に保管される座標を包含するスペースの定義

空間列の座標は、通常、地球の各部分にまたがるロケーションを定義します。東から西、北から南へと、その範囲が広がるスペースは*空間エクステント*と呼ばれます。例えば、座標が空間列に保管されている、洪水地帯の区域を考えてみます。これらの座標の最西端と最東端の経度値は、それぞれ、–24.556 と –19.338 であり、最北端と最南端の緯度値は、それぞれ、18.819 と 15.809 です。洪水地帯の空間エクステントは、2 つの緯度間の西-東平面と 2 つの経度間の北-南平面に及ぶスペースです。これらの値を特定のパラメーターに割り当てることによって、空間参照系に入れることができます。空間列に Z 座標の指標が含まれている場合は、空間参照系にも、最高と最低の Z 座標の指標を入れる必要があります。

空間エクステント という用語は、直前の段落でみたように、ロケーションの実際の範囲を示すだけでなく、潜在的なロケーションの範囲も示すことができます。前述の例の洪水地帯が、次の 5 年間で広がると仮定すると、5 年目の最後に、平面の最西端、最東端、最北端、最南端の座標がどうなっているかを算定することができます。次に、現行座標の代わりに、これらの算定値を、空間エクステント用のパラメーターに割り当てることができます。この方法により、洪水地帯が拡大して空間列に値の大きな緯度や経度が追加された場合でも、空間参照系を保持することができます。これとは異なり、空間参照系が元の緯度や経度に限定されている場合は、洪水地帯が大きくなったときに、空間参照系を変更するか、置き換える必要があります。

パフォーマンスを向上させる値への変換

通常、座標システムのほとんどの座標は小数値であり、一部は整数です。さらに、起点の東側の座標は正の値であり、西側の座標は負の値です。Spatial Extender に保管される前に、負の座標は正の値に変換され、小数の座標は整数の座標に変換されます。この結果、すべての座標は、Spatial Extender によって、正の整数として保管されます。このようにする理由は、座標を処理するときのパフォーマンスを高めることにあります。

前の段落で説明した変換を行うため、空間参照系の、いずれかのパラメーターが使用されます。*オフセット* と呼ばれるパラメーターを、負の各座標から差し引くと、正の値が残ります。小数の各座標に、 *スケール係数* と呼ばれるパラメーターをかけると、小数の座標と同じ精度の整数を得ることができます。 (オフセットは、負の座標からだけでなく、正の座標からも引かれます。また、スケール係数は、小数の座標だけでなく、小数以外の座標にも乗算されます。このようにすれば、正の非小数の座標を、負の小数の座標と同等のものとして扱うことができます。)

これらの変換は内部的に行われます。また、変換が有効なのは、座標が検索されるまでです。入力と照会の結果には、常に、元の変換されていないフォーマットの座標が含まれます。

デフォルトの空間参照系を使用するか新規システムを作成するかを決定する

使用する座標システムを決定すると、作業する座標データに適した空間参照系の準備ができます。 DB2 Spatial Extender は、空間データに対応する 5 つの空間参照系を、DB2 Geodetic Data Management Feature は、測地データに対応する 318 の測地参照系を提供しています。

デフォルトの空間参照系、あるいは定義済みの測地参照系に使用できるものがあるかどうかを判別するため、以下の質問に答えてください。

1. デフォルトの空間参照系のベースになる座標系に、作業対象の地域が含まれているか。これらの座標システムは、 49 ページの『DB2 Spatial Extender とともに提供される空間参照系』に示されています。
2. 地理座標系のデータが、「度 (10 進数を使用するもの)」あるいは「グラジアン」を測定単位として使用しているか。データが地球表面上の広い領域を網羅するものになっているか。距離、長さ、面積などを正確に算出する必要があるか。データが北極、南極、国際日付変更線の近くを網羅するものになっているか。答えが「イエス」になる質問が 1 つでもあれば、おそらく、318 の定義済み測地参照系のうちのいずれかを使用する必要があります。定義済み測地参照系の詳細については、 184 ページの『DB2 Geodetic Data Management Feature によってサポートされる測地系』を参照してください。
3. デフォルトの空間参照系に関連付けられている変換の係数は、扱おうとしている座標データに対しても有効か。

   Spatial Extender は、オフセット値とスケール係数を使用して、提供される座標データを正の整数に変換します。座標データが、特定のデフォルト空間参照系のオフセット値やスケール係数でも使用できるかどうかを判別するには、以下のことを行います。

   a. 51 ページの『座標データを整数にトランスフォームする変換係数』 に記載されている情報を確認します。
   b. これらの係数がデフォルト空間参照系でどのように定義されているかを調べる。オフセット値を最小 X-Y 座標に適用した後に、これらの座標がどちらも 0 より大きくならない場合は、新規の空間参照系を作成して自力でオフセットを定義する必要があります。新規の空間参照系の作成の詳細については、53 ページの『空間参照系の作成』 を参照してください。
4. 処理するデータに高さと深さの座標 (Z 座標) あるいは指標 (M 座標) が含まれているか。 Z 座標、あるいは M 座標が含まれている場合は、データに適合した Z あるいは M オフセット、およびスケール係数を備えた空間参照系を新たに作成する必要があります。
5. 既存の空間参照系、あるいは測地参照系がデータの処理に利用できない場合は、53 ページの『空間参照系の作成』を行う必要があります。

必要な空間参照系が決定できたら、以下のいずれかのタスクを行う際に、決定したシステムを Spatial Extender に指定してください。

- 『DB2 Spatial Extender とともに提供される空間参照系』
- 184 ページの『DB2 Geodetic Data Management Feature によってサポートされる測地系』

DB2 Spatial Extender とともに提供される空間参照系

空間参照系は座標データを正の整数に変換します。

下の表は、DB2 Spatial Extender が提供している空間参照系と、各空間参照系のベースになっている座標システム、および DB2 Spatial Extender が座標データを正の整数の値に変換する際に使用するオフセット値とスケール係数を示しています。この空間参照系についての情報は、DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビューで見ることができます。

小数の度数で作業を進める場合は、デフォルト空間参照系のオフセット値とスケール係数で、全範囲の経度・緯度座標がサポートされ、約 10 cm に相当する小数点以下 6 桁までの値が保持されます。Sample Spatial Map のデータとジオコーダー参照データは、小数の度数を使用します。

また、ジオコーダーを使用する計画がある場合、ジオコーダーは米国国内の住所にしか使えないため、 GCS_NORTH_AMERICAN_1983 座標システムなど、米国の座標を扱う空間参照系を選択または作成するようにしてください。どの座標システムから地理データを取り出すかを指定しなかった場合、Spatial Extenderは DEFAULT_SRS 空間参照系を使用します。

必要に適したデフォルトの空間参照系がない場合は、新規の空間参照系を作成できます。

表 2. DB2 Spatial Extender で提供されている空間参照系

空間参照系	SRS ID	座標システム	オフセット値	スケール係数	使用する場合
DEFAULT_SRS	0	なし	xOffset = 0 yOffset = 0 zOffset = 0 mOffset = 0	xScale = 1 yScale = 1 zScale = 1 mScale = 1	このシステムは、データが座標システムから独立していて、座標システムを指定できない、あるいは指定する必要がない場合に選択できます。

表 2. DB2 Spatial Extender で提供されている空間参照系 (続き)

空間参照系	SRS ID	座標システム	オフセット値	スケール係数	使用する場合
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	この空間参照系は、DB2 Spatial Extender に同梱されている米国のサンプル・データを使用する計画がある場合に選択できます。 処理する座標データが 1983 以降に収集されたものである場合は、NAD27_SRS_1002 ではなくこのシステムを使用してください。
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	この空間参照系は、DB2 Spatial Extender に同梱されている米国のサンプル・データを使用する計画がある場合に選択できます。 処理する座標データが 1983 以前に収集されたものである場合は、NAD83_SRS_1 ではなくこのシステムを使用してください。このシステムは、他のデフォルト空間参照系に比べて高い精度を持っています。

表 2. DB2 Spatial Extender で提供されている空間参照系 (続き)

空間参照系	SRS ID	座標システム	オフセット値	スケール係数	使用する場合
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	この空間参照系は、米国国外のデータを処理する場合に選択できます (このシステムは、世界中の座標を扱います)。 DB2 Spatial Extender に付属しているデフォルトのジオコーダーを使用する計画がある場合は、このシステムは使用しないでください (このジオコーダーは米国国内の住所にしか使用できません)。
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	これは、ドイツ国内の住所の座標システムをベースにした空間参照系です。

座標データを整数にトランスフォームする変換係数

DB2 Spatial Extender はオフセット値とスケール係数を使用して、提供される座標データを正の整数に変換します。 デフォルトの空間参照系は、あらかじめ対応するオフセット値やスケール係数を持っています。新たに空間参照系を作る場合は、データに適合するスケール係数（場合によってはオフセット値も）を指定します。

オフセット値

オフセット値というのは、剰余として正の値のみが残されるよう、すべての座標から差し引かれる数値のことです 。 Spatial Extender は以下の公式を使用して座標データを変換し、すべての座標値が調整されて正の数になるようにします。

注意 : 以下の公式で、『min』 という表記は『すべての値の最小値』を表します。例えば、『min(x)』 は 『すべての x 座標の最小値』を意味します。各地理ディメンションのオフセットは ディメンションOffset で表されます。例えば、xOffset はすべての X 座標に適用されるオフセット値です。

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

スケール係数

スケール係数とは、小数の座標および指標で乗算すると、元の座標および指標以上の有効桁数を持つ整数を生成する値のことです。 Spatial Extender は以下の公式を使用して小数座標データを変換し、すべての座標値が調整されて正の整数になるようにします。変換によって得ることができる値が、 2^{53} (およそ $9 * 10^{15}$) を超えることはできません。

注意 : 以下の公式で、 『max』 という表記は『すべての値の最大値』を表します。各地理ディメンションのオフセットは ディメンション Offset で表されます (例えば、xOffset はすべての X 座標に適用されるオフセット値です)。各地理ディメンションのスケール係数は ディメンション Scale で表されます (例えば、xScale はすべてのX 座標に適用されるスケール係数です)。

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

手持ちの座標データに最適のスケール係数を選択する場合、必ず以下のようにしてください。

- X および Y 座標に同じスケール係数を使用する。
- 小数の X 座標または小数の Y 座標で乗算されたときに、スケール係数が 2^{53} より小さくなるようにする。そのためには、スケール係数を 10 の累乗にするという方法がよく使われます。つまり、スケール係数を、10 の 1 乗 (10)、10 の 2 乗 (100)、10 の 3 乗 (1000) などにする (必要に応じて 4 乗以上も使う) ということです。
- スケール係数を十分に大きくし、新しい整数の有効桁数が、少なくとも元の小数座標と同じになるようにする。

例

例えば、ST_Point 関数が、X 座標 10.01、Y 座標 20.03、および空間参照系の ID から成る入力を受け取ったとします。ST_Point は呼び出されると、空間参照系の X および Y 座標のスケール係数によって、値 10.01 および値 20.03 を乗算します。このスケール係数が 10 である場合、Spatial Extender が保管する整数はそれぞれ 100 と 200 になります。これらの整数の有効桁数 (3) は座標の有効桁数 (4) よりも少ないので、DB2 Spatial Extender はこれらの整数を元の座標に変換し直したり、これらの座標が属する座標システムと整合性のある値を取り出したりすることはできません。しかしスケール係数が 100 である場合、DB2 Spatial Extender が保管する結果の整数は、1001 と 2003、つまり元の座標に変換し直せるか、または互換性のある座標を取り出せる値になります。

オフセット値およびスケール係数の単位

既存の空間参照システムを使用するか、新規のシステムを作成するかにかかわらず、オフセット値とスケール係数の単位は使用している座標システムのタイプによって左右されます。例えば、地理座標システムを使用している場合には、値の単位は「度 (10 進数を使用するもの)」などの角度単位になります。投影座標システムを使用している場合には、値の単位はメートルやフィートなどの線形単位になります。

空間参照系の作成

DB2 Spatial Extender に用意された空間参照系がどれも自分のデータに適合しない場合は、新規空間参照系を作成します。

このタスクは次のように行います。

1. インターフェースを選択します。

   以下のいずれかの方法によって、空間参照系を作成できます。

   - DB2 コントロール・センターの「空間参照系の作成」ウィンドウを使用する。このウィンドウの使用法に関する詳細は、オンライン・ヘルプを参照してください。
   - db2se コマンド行プロセッサーから db2se create_srs コマンドを発行する。
   - db2se.ST_create_srs ストアード・プロシージャーを呼び出すアプリケーションを実行する。

2. 適切な空間参照系 ID (SRID) を指定します。
   - 球形地球を基にした測地データの場合、200000318 から 2000001000 までの範囲 SRID 値を指定します。
   - 平面地球を基にした空間データの場合は、まだ定義されていない SRID を指定します。

3. 希望する精度を決定します。

   次のいずれかが可能です。

   - 作業を行いたい地域のエクステントと、座標データに使用するスケール係数を指定する。 Spatial Extender は、ユーザーが指定したエクステントを読み込み、オフセットを自動的に算出します。エクステントは、以下のいずれかの方法で指定できます。
     - コントロール・センターの「空間参照系の作成」ウィンドウで、「**エクステント** 」を選択する。
     - db2se create_srs コマンド、または db2se.ST_create_srs ストアード・プロシージャーに適したパラメーターを指定する。
   - オフセット値 (Spatial Extender が負の値を正の値に変換するために必要) とスケール係数 (Spatial Extender が小数値を整数に変換するのに必要) の両方を指定する。この方法は、高い正確性、精度が必要な場合に使用します。 オフセット値とスケール係数は、以下のいずれかの方法で指定できます。
     - コントロール・センターの「空間参照系の作成」ウィンドウで、「**オフセット**」を選択する。
     - db2se create_srs コマンド、または db2se.ST_create_srs ストアード・プロシージャーに適したパラメーターを指定する。

4. Spatial Extender が座標データを正の整数に変換するのに必要な変換情報を算出し、選択したインターフェースにこの情報を渡します。

   この情報は、ステップ 3 で選択したメソッドによって変わります。

   - ステップ 3 の「エクステント」メソッドを使用する場合は、次の情報を算出する必要があります。

- スケール係数。処理する座標の中に小数値が含まれている場合は、スケール係数を算出します。スケール係数とは、小数の座標および指標で乗算すると、元の座標および指標以上の有効桁数を持つ整数を生成する数値のことです。座標が整数である場合は、スケール係数は 1 に設定できます。座標が小数の値の場合は、スケール係数は、小数部分を整数値に変換する数に設定する必要があります。例えば、座標単位がメートルで、データの正確性が 1 cm の場合は、100 のスケール係数が必要になります。
- 座標と指標の最小および最大値。

• ステップ 3 の「オフセット」メソッドを使用する場合は、次の情報を算出する必要があります。
  - オフセット値

    座標データに負の数値や指標が含まれている場合は、希望するオフセット値を指定する必要があります。オフセットというのは、剰余として正の値のみが残されるよう、すべての座標から差し引かれる数値のことです 。正の値の座標を処理するときは、オフセット値をすべて 0 にしてください。処理する座標が正の値でないときは、座標データに適用したときの値が、最大の正の整数値より小さい整数になるようなオフセット値を設定してください (9,007,199,254,740,992) 。
  - スケール係数

    示そうとするロケーションの座標に小数値が含まれる場合は、使用するスケール係数を決定し、そのスケール係数を「空間参照系の作成」ウィンドウに入力します。

5. db2se create_srs コマンド、あるいは db2se.ST_create_srs ストアード・プロシージャーを実行して、ストアード・プロシージャーを作成します。

   例えば、以下のコマンドでは、mysrs という名前の空間参照系が作成されます。

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

スケール係数の計算

前提条件

空間参照系を作成する際、処理する座標の中に小数値が含まれている場合は、その座標や指標に対応する適切なスケール係数を算出する必要があります。スケール係数とは、小数の座標および指標で乗算すると、元の座標および指標以上の有効桁数を持つ整数を生成する数値のことです。

スケール係数を算出した後で、エクステント値を判別する必要があります。次に db2se create_srs コマンド、あるいは db2se.ST_create_srs ストアード・プロシージャーを実行します。

スケール係数を計算するには、次のようにします。

1. 小数の、または小数と思われる X 座標と Y 座標を判別します。たとえば、多数の X 座標と Y 座標を扱っており、そのうちの 3 つが小数であるとします (1.23、5.1235、および 6.789)。

2. 最も長精度の小数の座標を見つけます。次に、この座標に 10 の何乗を係数として乗算すると、精度の等しい整数になるか判別します。例えば、現在の例では、3 つの小数の座標のうち、5.1235 が最も長精度です。 10 の 4 乗 (10000) を乗算すると、整数の 51235 になります。
3. 上記の乗算によって求められる整数が、2^{53} より小さいかどうかを確認します。この場合、51235 は大きすぎるとは言えません。しかし、扱っている X 座標と Y 座標の中に、1.23、5.11235、および 6.789 以外にも、 4 つ目の小数として 10000000006.789876 があるとします。この座標の小数部の精度は他の 3 つの座標より長いので、この座標 (5.1235 ではない) に 10 の累乗を乗算します。この値を整数に変換するには、10 の 6 乗 (1000000) を乗算することになります。しかし、結果として得られる値、 10000000006789876 は、2^{53} より大きくなります。 DB2 Spatial Extender が格納しようとすると、結果は予測できないものになります。

   この問題を避けるには、10 の累乗のうち、元の座標に乗算するときに DB2 Spatial Extender で格納可能な整数に切り捨てる際に失われる小数の精度が最小になる値を選択します。この例の場合は、10 の 5 乗 (100000) を選択できます。 100000 を 10000000006.789876 に掛けると、 1000000000678987.6 になります。DB2 Spatial Extender は、この数値を丸め、正確性を少し下げて 1000000000678988 にします。

座標データを整数にトランスフォームする変換係数

DB2 Spatial Extender はオフセット値とスケール係数を使用して、提供される座標データを正の整数に変換します。 デフォルトの空間参照系は、あらかじめ対応するオフセット値やスケール係数を持っています。新たに空間参照系を作る場合は、データに適合するスケール係数（場合によってはオフセット値も）を指定します。

最小および最大の座標と指標を判別する

最小および最大の座標と指標の決定は、次の条件に当てはまる場合には以下のような手順で行います。

- DB2 Spatial Extender が提供する空間参照系がいずれも自分のデータに適合しないため、新たな空間参照系を作成しようとしている。
- 座標を変換するためにエクステント・トランスフォーメーションを使用しようとしている。

空間参照系を作成するに際してエクステント・トランスフォーメーションを指定する場合は、最小および最大の座標と指標を決定する必要があります。

エクステント値の決定後、小数値を含む座標がある場合は、そのスケール係数を計算する必要があります。小数値を含む座標がない場合、db2se create_srs コマンド、あるいは db2se.ST_create_srs ストアード・プロシージャーを実行します。

表示したいロケーションの座標と指標の最小および最大値を決定するには、次のようにします。

1. X 座標の最小および最大値を決定する。 最小 X 座標を見つけるには、範囲内で最も西側にある X 座標を識別します。 (ロケーションが起点の西側にある場

合は、この座標は負の値になります。) 最大 X 座標を見つけるには、範囲内で最も東側にある X 座標を識別します。例えば、油田を表示する場合、各油田が X 座標と Y 座標の組み合わせで定義されている場合は、最も西側にある油田のロケーションを示す X 座標が最小 X 座標に、最も東側にある油田のロケーションを示す X 座標が最大 X 座標になります。

ヒント: 複数ポリゴンのような複数地形タイプの場合は、計算している方角で最も遠くにあるポリゴンの最も遠い点を取るようにしてください。例えば、最小 X 座標を確認するときは、複数ポリゴンの中で最も西側にあるポリゴンの、最西端の X 座標を識別してください。

2. Y 座標の最小および最大値を決定する。 最小 Y 座標を見つけるには、範囲内で最も南側にある Y 座標を識別します。 (ロケーションが起点の南側にある場合は、この座標は負の値になります。) 最大 Y 座標を判別するには、範囲内で最も北側にある Y 座標を識別します。
3. Z 座標の最小および最大値を決定する。 最小 Z 座標は最も深い座標であり、最大 Z 座標は最も高い座標です。
4. 最小指標および最大指標を決定する。 空間データに指標を組み込む場合は、どの指標の数値が最も高く、どの指標の数値が最も低いかを判別します。

オフセット値の計算

座標データに負の数値や指標が含まれている場合は、オフセット値を指定します。

空間参照系を作成する際、座標データに負の数値や指標が含まれている場合は、適切なオフセット値を指定する必要があります。オフセットというのは、剰余として正の値のみが残されるよう、すべての座標から差し引かれる数値のことです 。座標の数値や指標を負でなく、正の整数にすれば、空間操作のパフォーマンスを向上することができます。

処理する座標のオフセット値は、次のようにして計算します。

1. 表すロケーションの座標範囲のうちで、最も小さいの負の X、Y、および Z 座標を判別します。データに負の指標が含まれる場合は、これらの指標の最小値を判別します。
2. 推奨のオプション: 取り扱うロケーションを包含する範囲は実際の範囲より大きいことを、DB2 Spatial Extender に示します。これにより、これらのロケーションに関するデータを空間列に書き込むと、新しい地形のロケーションに関するデータが範囲の外側に追加されたときに、使用している空間参照系を別のものに置き換えなくても、そのデータを追加できるようになります。

   ステップ 1 で判別したそれぞれの座標と指標に対して、座標や指標の 5% から 10% に相当する値を追加します。このようにして生成された値を、*増補値* と呼びます。例えば、最小の負の X 座標が –100 の場合、その値に –5 を加算できます。このときに生成される増補は –105 になります。後で空間参照系を作成するときに、最小の X 座標は、実際の値の –100 でなく、–105 であることを指示します。これによって、DB2 Spatial Extender は、範囲の最西端の限界が –105 であると解釈します。

3. X 軸の増補値から差し引くとゼロになる値を見つけてください。その値が、X 座標のオフセット値です。 DB2 Spatial Extender は、すべての X 座標からこの数を差し引き、正の値のみを示します。

   例えば、増補 X 値が –105 の場合、計算結果を 0 にするにはこの値から –105 を減算する必要があります。すると DB2 Spatial Extender により、表そうとしている地形に関連したすべての X 座標から –105 が減算されます。これらの座標はすべて –100 以下であるため、減算結果の値はすべて正の数になります。
4. 増補 Y 値、増補 Z 値、増補指標のそれぞれについてステップ 3 を繰り返します。

空間参照系の作成

DB2 Spatial Extender に用意された空間参照系がどれも自分のデータに適合しない場合は、新規空間参照系を作成します。

このタスクは次のように行います。

1. インターフェースを選択します。

   以下のいずれかの方法によって、空間参照系を作成できます。

   - DB2 コントロール・センターの「空間参照系の作成」ウィンドウを使用する。このウィンドウの使用法に関する詳細は、オンライン・ヘルプを参照してください。
   - db2se コマンド行プロセッサーから db2se create_srs コマンドを発行する。
   - db2se.ST_create_srs ストアード・プロシージャーを呼び出すアプリケーションを実行する。
2. 適切な空間参照系 ID (SRID) を指定します。
   - 球形地球を基にした測地データの場合、200000318 から 2000001000 までの範囲 SRID 値を指定します。
   - 平面地球を基にした空間データの場合は、まだ定義されていない SRID を指定します。
3. 希望する精度を決定します。

   次のいずれかが可能です。

   - 作業を行いたい地域のエクステントと、座標データに使用するスケール係数を指定する。 Spatial Extender は、ユーザーが指定したエクステントを読み込み、オフセットを自動的に算出します。エクステントは、以下のいずれかの方法で指定できます。
     - コントロール・センターの「空間参照系の作成」ウィンドウで、「**エクステント** 」を選択する。
     - db2se create_srs コマンド、または db2se.ST_create_srs ストアード・プロシージャーに適したパラメーターを指定する。
   - オフセット値 (Spatial Extender が負の値を正の値に変換するために必要) とスケール係数 (Spatial Extender が小数値を整数に変換するのに必要) の両方を指定する。この方法は、高い正確性、精度が必要な場合に使用します。 オフセット値とスケール係数は、以下のいずれかの方法で指定できます。

– コントロール・センターの「空間参照系の作成」ウィンドウで、「**オフセット**」を選択する。
– db2se create_srs コマンド、または db2se.ST_create_srs ストアード・プロシージャーに適したパラメーターを指定する。

4. Spatial Extender が座標データを正の整数に変換するのに必要な変換情報を算出し、選択したインターフェースにこの情報を渡します。

   この情報は、ステップ 3 で選択したメソッドによって変わります。

   • ステップ 3 の「エクステント」メソッドを使用する場合は、次の情報を算出する必要があります。
     – スケール係数。処理する座標の中に小数値が含まれている場合は、スケール係数を算出します。スケール係数とは、小数の座標および指標で乗算すると、元の座標および指標以上の有効桁数を持つ整数を生成する数値のことです。座標が整数である場合は、スケール係数は 1 に設定できます。座標が小数の値の場合は、スケール係数は、小数部分を整数値に変換する数に設定する必要があります。例えば、座標単位がメートルで、データの正確性が 1 cm の場合は、100 のスケール係数が必要になります。
     – 座標と指標の最小および最大値。
   • ステップ 3 の「オフセット」メソッドを使用する場合は、次の情報を算出する必要があります。
     – オフセット値

       座標データに負の数値や指標が含まれている場合は、希望するオフセット値を指定する必要があります。オフセットというのは、剰余として正の値のみが残されるよう、すべての座標から差し引かれる数値のことです 。正の値の座標を処理するときは、オフセット値をすべて 0 にしてください。処理する座標が正の値でないときは、座標データに適用したときの値が、最大の正の整数値より小さい整数になるようなオフセット値を設定してください (9,007,199,254,740,992) 。
     – スケール係数

       示そうとするロケーションの座標に小数値が含まれる場合は、使用するスケール係数を決定し、そのスケール係数を「空間参照系の作成」ウィンドウに入力します。

5. db2se create_srs コマンド、あるいは db2se.ST_create_srs ストアード・プロシージャーを実行して、ストアード・プロシージャーを作成します。

   例えば、以下のコマンドでは、mysrs という名前の空間参照系が作成されます。

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

第 9 章 空間列のセットアップ

プロジェクトのための空間データの入手を準備する作業には、座標システムおよび空間参照系を選択または作成する作業の他に、データを入れる表の列 (1 つまたは複数) を指定する作業があります。この章では、以下の項目を説明しています。

- 列の照会結果をグラフィカルに表示する方法および、列のデータ・タイプを選択するためのガイドライン
- 列を指定するタスク
- 列の内容をグラフィック形式で表示できるツールが、その列をアクセスできるようにするためのタスク

空間列

表示可能な内容をもった空間列

空間列を照会するのに ArcExplorer for DB2® などの視覚化ツールを使用すると、区画境界地図や道路システムのレイアウトなどのように、結果がグラフィカル表示で戻ります。 視覚化ツールのなかには、列のすべての行で、同じ空間参照系を使用する必要のあるものもあります。この制約に従わせるには、空間参照系にその列を登録してください。

空間データ・タイプ

空間操作にデータベースを使用できるようにすると、DB2 Spatial Extender によりデータベースに構造化データ・タイプの階層が提供されます。

60 ページの図 12 はこの階層を示しています。この図で、インスタンス化可能なタイプは背景が白くなっており、インスタンス化不可能なタイプは背景に陰影が付いています。

インスタンス化可能なデータ・タイプは、 ST_Point、ST_LineString、ST_Polygon、ST_GeomCollection、ST_MultiPoint、ST_MultiPolygon、ST_MultiLineString です。

インスタンス化不可能なデータ・タイプは、ST_Geometry、ST_Curve、ST_Surface、ST_MultiSurface、および ST_MultiCurve です。

© Copyright IBM Corp. 1998, 2009

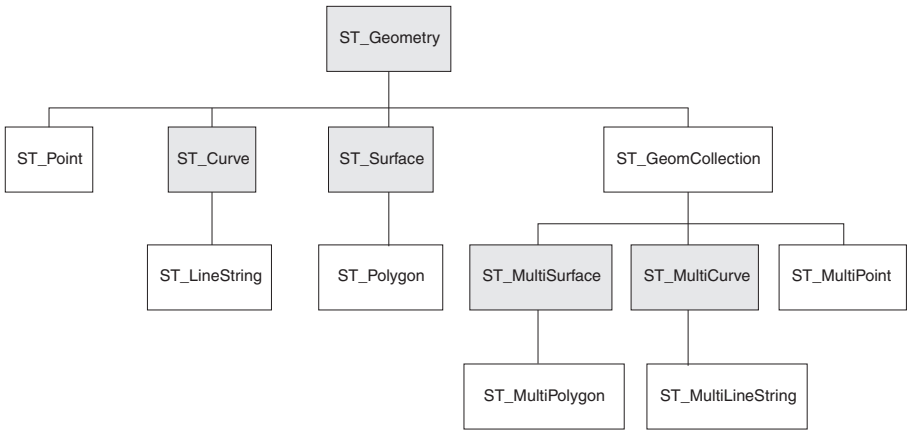


図 12. 空間データの階層： 白いボックスに名前のあるデータ・タイプはインスタンス化可能。陰影が付いているボックスに名前のあるデータ・タイプはインスタンス化不可能。

図 12 の階層には、以下のデータ・タイプが含まれています。

- 単一のユニットから成っていると見なすことのできる地形のデータ・タイプ。例えば、個人住居の敷地や孤立した湖など。
- 複数のユニットまたはコンポーネントから成る地形のデータ・タイプ。例えば、運河システムや湖にある島々など。
- すべての種類の地形のデータ・タイプ。

単一単位の地形データ・タイプ

ST_Point、ST_LineString、および ST_Polygon を使用して、単一の単位から成ると見なすことのできる地形によって占められるスペースを定義した座標を格納します。

- ST_Point を使用して、孤立した地形によって占められるスペースの位置を指示します。この種の地形は非常に小さい場合 (井戸など) や、非常に大きい場合 (街など) や、その中間の大きさの場合 (団地や公園など) があります。いずれの場合でも、スペースを示す地点は、東西に走る座標軸 (緯線など) と南北に走る座標軸 (経線など) の交点にすることができます。 ST_Point データ項目には、この交点を定義した X 座標と Y 座標が含まれます。X 座標は東西に走る線上の交点を示し、Y 座標は南北に走る線上の交点を示します。
- ST_Linestring は、線状の地形 (道路、水路、配管など) によって占められるスペースを定義する座標用に使用します。
- ST_Polygon は、ポリゴンにより表現されるスペースのエクステント (選挙区、森、野生生物の生息地など) を示す場合に使用します。ST_Polygon データ項目は、このような地形の境界線を定義した座標群から成ります。

ST_Polygon と ST_Point を同じ地形に使用できる場合があります。例えば、団地に関する空間が必要であるとします。団地内の個々の建物があるスペースの位置を表したい場合は、ST_Point を使用して、個々の地点を定義した X 座標と Y 座標を格納します。他方、団地全体が占有する区域を表したい場合は、ST_Polygon を使用して、この区域の境界線を定義する座標を格納します。

複数のユニットから成る地形のデータ・タイプ

ST_MultiPoint、ST_MultiLineString、および ST_MultiPolygon を使用して、複数のユニットから成る地形によって占められるスペースを定義する座標を格納します。

- ST_MultiPoint を使用して、それぞれのロケーションが X 座標と Y 座標によって示される、複数のユニットで構成されている地形を表します。例えば、ある表の行が群島を表すとします。各島の X 座標と Y 座標は確定されています。表に、これらの座標と群島の座標全体を入れる場合は、ST_MultiPoint 列がこれらの座標を保持するように定義します。
- ST_MultiLineString を使用して、複数の線状ユニットから成る地形を表し、これらのユニットのロケーションと各地形のロケーションを示す全体の座標を格納します。例えば、ある表の行が水系を表すとします。表に、これらの水系とそのコンポーネントのロケーションを示す座標を入れる場合は、ST_MultiLineString 列がこれらの座標を保持するように定義します。
- ST_MultiPolygon を使用して、複数のポリゴンから成る地形を表し、それぞれのユニットのロケーションと各地形を全体としてみたロケーションを示す座標を格納します。例えば、ある表の行が地方の郡とそれぞれの郡にある農場を表すとします。表に、郡と農場のロケーションを示す座標を入れる場合には、ST_MultiPolygon 列がこれらの座標を保持するように定義します。

複数ユニットは、個々のエンティティーの集合を意味するものではありません。そうではなく、複数ユニットは全体を構成する部分の集約を指します。

すべての地形のデータ・タイプ

どのデータ・タイプを使用するか分からない場合は、ST_Geometry を使用できます。

ST_Geometry は、他のデータ・タイプが属する階層のルートなので、ST_Geometry 列には、他のデータ・タイプの列に入れることができるのと同じ種類のデータ項目を入れることができます。

重要: 提供されている DB2SE_USA_GEOCODER を使用して、空間列にデータを生成する場合は、列のタイプは ST_Point または ST_Geometry であることが必要です。ただし、ある種の視覚化ツールでは、ST_Geometry 列をサポートせず、ST_Geometry の適切なサブタイプが割り当てられた列しかサポートしないものがあります。

空間列の作成

空間データの保管と検索を行うには、空間列を作成する必要があります。

空間列を作成するときは、DB2 SQL CREATE TABLE ステートメント、あるいは ALTER TABLE ステートメントに必要とされる権限を持ったユーザー ID が必要です。ユーザー ID には、次の権限、あるいは特権のうちの少なくとも 1 つが必要です。

- 列を含む表が入っているデータベースに関する DBADM 権限
- データベースに対する CREATETAB 権限および表スペースに対する USE 特権に加えて、以下のいずれか。
  - 索引のスキーマが存在しない場合は、データベースに対する IMPLICIT_SCHEMA 権限
  - 索引のスキーマ名が既存のスキーマを参照する場合は、そのスキーマに対する CREATEIN 特権

- 変更する表に対する ALTER 特権
- 変更する表に対する CONTROL 特権
- 変更する表のスキーマに対する ALTERIN 特権

このタスクは、「プロジェクトのための空間情報をセットアップする」というタスクの一部です。座標系を選択し、自分のデータにどの空間参照系を使用するかを決めた後、既存の表に空間列を作成するか、新しい表に空間データをインポートします。

このタスクは次のように行います。

データベースに空間列を作成するには、次のうちのいずれかの方法を採ります。

- DB2 の CREATE TABLE ステートメントを使用して表を作成し、その表に空間列を組み込む。
- DB2 の ALTER TABLE ステートメントを使用して、空間列を既存の表に加える。
- DB2 コントロール・センターの 「空間列の作成 (Create Spatial Column)」ウィンドウを使用する。表から「空間列 (Spatial Columns)」ウィンドウを開く。
- シェイプ・ファイルから空間データをインポートする場合は、DB2 Spatial Extender を使用して表を作成し、この表にそのデータを保持する列を作成する。「形状データを新規の表または既存の表にインポートする」を参照。

空間リソースのセットアップでの次のタスクは、「空間列の登録」です。

空間列の登録

空間列を登録すると、可能な場合、すべての形状が指定された空間参照系を必ず使用するように、表に制約が作成されます。

前提条件

以下の場合には、空間列を登録しなければならなくなる可能性があります。

- 視覚化ツールによるアクセス

  空間列のデータをグラフィカルに表示する ArcExplorer for DB2 などの特定の視覚化ツールが必要な場合は、列のデータの整合性を確保する必要があります。これを行うには、列のすべての行で、同じ空間参照系が使用されなければならないという制約を適用します。この制約を適用するには、列を登録し、列の名前とその列に適用される空間参照系の両方を指定します。
- 空間インデックスによるアクセス

  空間インデックスが確実に正しい結果を戻すようにするため、索引を作成する必要のある空間列中のすべてのデータについて同じ座標システムを使用します。空間列は、すべてのデータで必ず同じ空間参照系と、同じ座標システムが使用されるよう登録します。

第 10 章 空間列にデータを入れる

空間列を作成し、表示ツールがアクセスする列を登録すれば、列に空間データを入れる準備ができたことになります。データを入れるには 3 つの方法があります。それは、データをインポートする方法、ジオコーダーを使用してビジネス・データからデータを得る方法、または空間処理関数を使用して、データを作成またはビジネス・データや他の空間データからデータを得る方法です。

空間データのインポートとエクスポートについて

DB2® Spatial Extender を使用して、ご使用のデータベースと外部データ・ソースとの間で、空間データを交換することができます。 より正確には、データ交換ファイルと呼ばれるファイルの形で、空間データをご使用のデータベースに転送することによって、外部ソースから空間データをインポートできます。ご使用のデータベースから空間データをデータ交換ファイルにエクスポートし、外部ソースはこのファイルから空間データを取り出すようにもできます。このセクションでは、空間データのインポートとエクスポートを行う理由の一部を紹介し、DB2 Spatial Extender がサポートするデータ交換ファイルの性質についても説明します。

空間データのインポートとエクスポートを行う理由

空間データをインポートすることにより、既に業界で使用可能になっている大量の空間を取り込むことができます。空間データをエクスポートすることにより、既存アプリケーションが、そのデータを標準のファイル・フォーマットで使用できるようになります。以下のシナリオを考えてみます。

- データベースの中に、販売オフィス、顧客、その他の業務関連事項を表す空間データが含まれている場合。このデータを、会社を取り巻く環境 (都市、道路、重要な場所など) を示す空間データで補足します。必要なデータは、地図のベンダーから入手できます。DB2 Spatial Extender を使えば、ベンダーが提供するデータ交換ファイルからデータをインポートできます。
- 空間データを Oracle システムから DB2 環境にマイグレーションする場合。Oracle ユーティリティーを使って、データをデータ交換ファイルに書き込みます。次に、DB2 Spatial Extender を使って、このファイルから、データを空間操作を使用可能にしたデータベースにインポートします。
- DB2 に接続しておらず、ジオブラウザーを使って、空間を顧客にビジュアルに表示する場合。 ブラウザーには、作業元となるファイルだけが必要です。データベースに接続する必要はありません。DB2 Spatial Extender を使えば、データをデータ交換ファイルにエクスポートしてから、ブラウザーを使ってデータをビジュアルに表示できます。

シェイプ・ファイル

DB2 Spatial Extender は、データ交換用のシェイプ・ファイルをサポートしています。シェイプ・ファイル は、ファイル名は同じであるがファイル拡張子の異なるフ

© Copyright IBM Corp. 1998, 2009

ファイルの集合を指します。この集合には、最高 4 つのファイルを含むことができます。それらの関数とは、以下のものです。

- ESRI が開発した事実上の業界標準フォーマットである、形状フォーマット の空間データが入っているファイル。このようなデータは、多くの場合、形状データと呼ばれます。形状データが入ったファイルの拡張子は .shp です。
- 形状データによって定義されるロケーションに関連したビジネス・データが入っているファイル。このファイルの拡張子は .dbf です。
- 形状データに対する索引が入っているファイル。このファイルの拡張子は .shx です。
- .shp ファイル中のデータの基となる座標システム仕様が入っているファイル。このファイルの拡張子は .prj です。

形状ファイルは多くの場合、ファイル・システムで生成されるデータをインポートするためと、ファイル・システム内のファイルにデータをエクスポートするために使用します。

DB2 Spatial Extender を使用して形状データをインポートする場合は、少なくとも 1 つの .shp ファイルを受け取ります。 ほとんどの場合、他の 3 種類の形状ファイルのうちの 1 つまたは複数のファイルも受け取ります。

空間データのインポート

ここでは、形状データおよび SDE 転送データをデータベースにインポートするタスクの概要を説明します。このセクションには、これらのタスクを実行するために知る必要のある細目 (例えば、プロセスとパラメーター) の相互参照が含まれています。

形状データを新規の表または既存の表にインポートする

形状データを既存の表またはビューにインポートしたり、1 回の操作で、表を作成し、この表に形状データをインポートしたりできます。

既存の表やビューに形状データをインポートするには、ユーザー ID は次の権限または特権の 1 つを持つ必要があります。

- 表またはビューが入ったデータベースに関する DATAACCESS 権限
- 表またはビューに関する CONTROL 特権
- 表またはビューに関する INSERT 特権
- 表またはビューに関する SELECT 特権 (表に ID 列でない ID 列がある場合だけ必要)

さらに以下のいずれかの特権が必要です。

- 入力ファイルやエラー・ファイルが入るディレクトリーへのアクセス権
- 入力ファイルに関する読み取り特権とエラー・ファイルに関する書き込み特権

表を自動的に作成し、形状データをこの表にインポートするには、ユーザー ID が次の権限または特権の 1 つを持つ必要があります。

- 表が入ったデータベースに関する DBADM 権限
- 表が入ったデータベースに関する CREATETAB 権限

- スキーマが既に存在する場合は、表が属するスキーマに関する CREATEIN 特権
- 表に指定したスキーマが存在しない場合、その表が含まれるデータベースに対する IMPLICIT_SCHEMA 権限

さらに以下のいずれかの特権が必要です。

- 入力ファイルやエラー・ファイルが入るディレクトリーへのアクセス権
- 入力ファイルに関する読み取り特権とエラー・ファイルに関する書き込み特権

以下のいずれかの方法を選択して、形状データをインポートします。

- 既存の表、既存の更新可能ビュー、または INSERT 用の INSTEAD OF トリガーが定義されている既存のビューの空間列に、形状データをインポートする。
- 自動的に、空間列をもった表を作成し、形状データをこの列にインポートする。

このタスクは次のように行います。

形状データのインポートに使用する方法を選択します。

- DB2 コントロール・センターの「シェイプ・データのインポート (Import Shape Data)」ウィンドウを使用する。
- db2se import_shape コマンドを発行する。
- db2gse.ST_import_shape ストアード・プロシージャーを呼び出すアプリケーションを実行する。

空間データのエクスポート

ここでは、形状ファイルおよび SDE 転送ファイルに空間データをエクスポートするタスクの概要を説明します。このセクションには、これらのタスクを実行するために知る必要のある細目 (例えば、プロセスとパラメーター) の相互参照が含まれています。

データを形状ファイルにエクスポートする

照会結果として戻された空間データを、形状ファイルにエクスポートできます。

形状ファイルにデータをエクスポートするには、ユーザー ID が次の特権をもっている必要があります。

- エクスポートしようとする結果を戻す副選択を実行するための特権
- データのエクスポート先のファイルが置かれるディレクトリーに書き込むための特権
- エクスポートされるデータを含むファイルを作成するための特権 (そのようなファイルがまだ存在しない場合に必要)

これらの特権とそれらの取得方法については、データベース管理者にお問い合わせください。

形状ファイルにエクスポートする空間データは、基本表、複数の表の結合または共用体、ビューを照会したときに戻される結果セット、空間処理関数の出力などのソースから得られる可能性があります。

データをエクスポートしようとするファイルが存在する場合には、DB2 Spatial Extender はデータをこのファイルに追加します。このようなファイルが存在しない場合には、DB2 Spatial Extender を使用してそのファイルを作成できます。

このタスクは次のように行います。

形状ファイルにデータをエクスポートする方法を選択します。

- DB2 コントロール・センターの「シェイプ・ファイルのエクスポート (Export Shape File)」ウィンドウからエクスポートを開始する。
- db2se コマンド行プロセッサーから、db2se export_shape コマンドを発行する。
- db2gse.ST_export_shape ストアード・プロシージャーを呼び出すアプリケーションを実行する。

ジオコーダーの使用法

このセクションでは、ジオコーディングの概念および次のタスクを紹介します。

- ジオコーダーに発行させようとする作業を定義する。例えば、ジオコーダーがいくつレコードを処理するたびにコミットを発行するかを指定します。
- データが表に追加されるか表の中で更新されたならばすぐにデータをジオコーディングするように、ジオコーダーをセットアップする。
- ジオコーダーをバッチ・モードで実行する

ジオコーダーとジオコーディング

ジオコーダー とジオコーディング という用語は、いくつかのコンテキストで使用されます。ここでは、この用語が出てきたときにその意味を明確に理解できるように、これらのコンテキストを選び出して説明します。この説明では、ジオコーダーとジオコーディング の定義、ジオコーダーが動作するモードとジオコーディングが行うより幅広い機能の説明、およびジオコーディングに関連するユーザーのタスクの要約を行います。

DB2® Spatial Extender では、ジオコーダーは、既存データ (関数の入力) を空間用語で理解できるデータ (関数の出力) に変換するスカラー関数です。通常、既存データは、ロケーションを説明したり、ロケーションに名前を付けたりするリレーショナル・データです。例えば、DB2 Spatial Extender と一緒に出荷されるジオコーダー DB2SE_USA_GEOCODER は、米国のアドレスを ST_Point データに変換します。DB2 Spatial Extender は、ベンダーやユーザーが提供するジオコーダーもサポートするので、それらのジオコーダーの入出力は、DB2SE_USA_GEOCODER と同様である必要はありません。 例をあげれば、ベンダー提供のジオコーダーの中には、アドレスを、DB2 は保管しないが、ファイルには書き出せる座標に変換するものがあります。 また、商業ビル内のオフィスの番号を、ビル内のオフィスのロケーションを定義する座標に変換したり、倉庫内の棚の ID を倉庫内の棚のロケーションを定義する座標に変換したりできるものもあります。

その他の場合では、ジオコーダーが変換する既存データは空間データである場合もあります。例えば、ユーザー提供のジオコーダーは、X 座標と Y 座標を、DB2 Spatial Extender のどれかのデータ・タイプに準拠するデータに変換する場合があります。

DB2 Spatial Extender では、ジオコーディング とは単に、ジオコーダーが入力を出力に変換する (例えば、アドレスを座標に変換する) 操作のことを意味します。

モード

ジオコーダーは、以下の 2 つのモードで実行されます。

- バッチ・モード では、ジオコーダーは単一の表からのすべての入力を、1 回の操作で変換しようとします。例えば、バッチ・モードでは、DB2SE_USA_GEOCODER は、1 つの表のすべてのアドレス (または、表の指定した行からなるサブセットにあるすべてのアドレス) を変換しようとします。
- 自動モード では、ジオコーダーは、データが表に挿入されるか、表の中で更新されるとすぐにデータを変換します。ジオコーダーは、表で定義されている INSERT トリガーや UPDATE トリガーによってアクティブ化されます。

ジオコーディング処理

ジオコーディングとは、他のデータから DB2 表にある空間列の説明を導き出すいくつかの操作のうちの 1 つの操作です。 ここでは、これらの操作全体をジオコーディング処理 と呼ぶことにします。ジオコーディング処理は、ジオコーダーによって異なります。例えば、DB2SE_USA_GEOCODER は、既知のアドレス・ファイルを検索して、入力として受け取った各アドレスが、既知のアドレスと指定された度合いで一致するかどうか調べます。既知のアドレスとは調査の際に検索する参照資料のようなものなので、これらのアドレス全体は、参照データ と呼ばれます。参照データを必要としないジオコーダーもあります。これらのジオコーダーでは、別の方法で入力が検査されます。DB2SE_USA_GEOCODER が行うジオコーディング処理は、次のとおりです。

1. DB2SE_USA_GEOCODER は、次のような操作を実行するように設計されています。
   a. DB2SE_USA_GEOCODER は、入力として受け取る各アドレスを解析します。
   b. DB2SE_USA_GEOCODER は、参照データを検索して、解析したアドレスにある道路名と、特定の度合いで類似する道路名を探します。この検索は、アドレスの郵便番号によって限定された区域内の道路に限定されます。
   c. 検索が成功すると、DB2SE_USA_GEOCODER は、見つけ出した道路上のアドレスの中に、解析したアドレスに特定の度合いで一致するものがあるかどうかを調べます。
   d. 一致するものを見つけると、DB2SE_USA_GEOCODER は、解析したアドレスをジオコーディングします。見つからない場合は、NULL を戻します。
2. DB2SE_USA_GEOCODER が解析したアドレスをジオコーディングすると、DB2 は、生成された座標を指定された空間列の中に入れます。
3. バッチ・モードで DB2SE_USA_GEOCODER がジオコーディングする場合には、DB2 Spatial Extender は、(a) DB2SE_USA_GEOCODER が入力レコードの処理を特定の回数終えるごとに、あるいは (b) DB2SE_USA_GEOCODER がすべての入力の処理を終えた後に、コミットを出します。

ユーザーのタスク

DB2 Spatial Extender では、ジオコーディングに関するタスクは次のとおりです。

- 指定する空間列について、特定のジオコーディング処理をどのように実行するかを定める。例えば、入力レコードの道路名と参照データの道路名とが一致すると見なす最低の度合いを設定すること。入力レコードのアドレスと参照データのアドレスとが一致すると見なす最低の度合いを設定すること。各コミットまでに、何個のレコードを処理するかを決定すること、などです。このタスクは、ジオコーディングのセットアップ またはジオコーディング操作のセットアップ と呼ぶことができます。
- データが表に追加されるか、表の中で更新されるたびに、データを自動的にジオコーディングするように指定すること。自動ジオコーディングが行われる場合、ジオコーディング操作のセットアップでユーザーが指定した指示が有効になります (コミットに関連した指示は例外であり、バッチ・ジオコーディングだけに適用されます)。このタスクは、自動的に実行されるジオコーダーのセットアップと呼ばれます。
- ジオコーダーをバッチ・モードで実行する。ユーザーが既にジオコーディング操作をセットアップしている場合には、ユーザーの指示がオーバーライドされない限り、各バッチ・セッションで有効のままです。セッション前にジオコーディング操作をセットアップしていない場合には、ユーザーは、その特定のセッションに対してそれらの操作をセットアップし、有効になるように指定できます。このタスクは、バッチ・モードでのジオコーダーの実行 およびバッチ・モードでのジオコーディングの実行 と呼ぶことができます。

ジオコーディング操作のセットアップ

DB2 Spatial Extender を使用すると、ジオコーダーを呼び出すときに行う必要のある作業を、前もって設定できます。

特定のジオコーダーに対して、ジオコーディング操作を設定するには、ユーザー ID が次のどれかの権限または特権をもっている必要があります。

- ジオコーダーが操作する表が入ったデータベースに関する DATAACCESS 権限。
- ジオコーダーが操作する各表に関する CONTROL 特権。
- ジオコーダーが操作する各表に関する SELECT 特権、および UPDATE 特権。

ジオコーダーの呼び出し時に以下のパラメーターを指定できます。

- ジオコーダーがどの列に対してデータを提供するか。
- ジオコーダーが表やビューから読み取る入力を、表やビューにある行のサブセットに限定するかどうか。
- バッチ・セッションで、ジオコーダーが 1 つの作業単位内でジオコーディングするレコードの範囲または数。
- ジオコーダーに固有の操作に関する要件。例えば、DB2SE_USA_GEOCODER は、参照データ中の対応レコードに指定する度合いか、それより高い度合いで一致するレコードしかジオコーディングできないことです。この度合いは、最小一致スコア と呼ばれます。

ジオコーダーを自動モードで実行するようにセットアップする前に、上記のリストのパラメーターを指定しておく必要があります。これ以後は、ジオコーダーが呼び出されるたびに (自動実行だけでなく、バッチ実行でも)、ジオコーディング操作は指定に従って実行されるようになります。例えば、作業単位内で 45 レコードをバッチ・モードでジオコーディングする指定を行った場合、45 レコードがジオコーデ

ィングされるたびに、コミットが発行されます。(例外: バッチ・ジオコーディングの個々のセッションで、指定をオーバーライドすることができます。)

ジオコーダーをバッチ・モードで実行する前に、ジオコーディング操作についてのデフォルトを設定する必要はありません。バッチ・セッションを開始するときに、その間どのように操作を実行するかを指定できるからです。バッチ・セッションでデフォルトを設定していても、必要に応じて、個々のセッションでそれらのデフォルトをオーバーライドできます。

このタスクは次のように行います。

ジオコーディングの操作をセットアップする方法を選択します。

- DB2 コントロール・センターの「ジオコーディングのセットアップ」ウィンドウからジオコーダーを呼び出す。
- db2se setup_gc コマンドを出す。
- db2gse.ST_setup_geocoding ストアード・プロシージャーを呼び出すアプリケーションを実行する。

推奨事項: DB2SE_USA_GEOCODER は住所データのレコードを読み取る際、そのレコードを参照データ内の対応レコードと突き合わせます。この処理方法の概要は次のとおりです。まず、参照データを検索して、レコードの郵便番号と同じ郵便番号をもつ番地を探します。特定の最小度合いか、これより高い度合いで、レコードの番地名に類似する番地名が見つかると、次に、全体の住所を探し始めます。特定の最小度合いか、これより高い度合いで、レコードの全体の住所に類似する全体の住所が見つかると、そのレコードをジオコーディングします。そのような住所が見つからない場合は、 NULL を戻します。

番地名が一致する必要のある最小度合いは、スペリング感度と呼ばれます。全体の住所が一致する必要のある最小度合いは、最小一致スコア と呼ばれます。例えば、スペリング感度が 80 の場合は、ジオコーダーが全体の住所を検索できるようになるには、番地名間の一致度合いが 80% 以上であることが必要です。最小一致スコアが 60 の場合は、ジオコーダーがレコードをジオコーディングできるようになるには、住所間の一致度合いが 60% 以上であることが必要です。

スペリング感度と最小一致スコアの値は指定できます。それらの値は調整しなければならない場合もあるので注意してください。例えば、スペリング感度と最小一致スコアの両方が 95 であると仮定します。ジオコーディングしようとする住所が注意深く検査されていないときは、95% の精度で一致するのは、非常にまれな場合です。この結果、ジオコーダーがこれらのレコードを処理する場合、NULL を戻す可能性が高くなります。このような場合は、スペリング感度と最小一致スコアを低くして、ジオコーダーをもう一度実行してください。スペリング感度と最小一致スコアの推奨値は、それぞれ、70 と 60 です。

この説明の始めのところで述べたように、ジオコーダーが表やビューから読み取る入力を、表やビューにある行のサブセットに限定するかどうかを決めることができます。例えば、以下のシナリオを考えてみます。

- バッチ・モードで表の住所をジオコーディングするために、ジオコーダーを呼び出します。残念ながら、最小一致スコアが高すぎるために、ジオコーダーがほとんどの住所を処理したときに、NULL を戻します。もう一度ジオコーダーを実行

するときには、最小一致スコアを低くします。入力をジオコーディングされなかった住所に制限するために、ジオコーダーが以前戻した NULL を含む行だけを、ジオコーダーが選択するように指定します。
- ジオコーダーは、特定の日付以後に追加された行だけを選択する。
- ジオコーダーは、特定の区域の住所を含む行だけを選択する。例えば、地域や州のブロックなどです。

この説明の最初のところで述べたように、バッチ・セッションで、ジオコーダーがジオコーディングする作業単位内のレコード数を決めることができます。ジオコーダーが、各作業単位で同数のレコードを処理するようにできます。あるいは、ジオコーダーが、1 つの作業単位で、表のすべてのレコードを処理するようにもできます。後者を選択する場合は、次のことに注意してください。
- 前者の場合に比べて、作業単位のサイズを制御しにくい。この結果、ジオコーダーが操作を行うときに、保持するロックの個数や作成するログ項目の個数を制御できなくなります。
- ロールバックを必要とするエラーが発生した場合には、ジオコーダーをすべてのレコードに対してもう一度実行する必要があります。表が非常に大きく、ほとんどのレコードが処理されたあとでエラーおよびロールバックが発生すると、リソースにかかるコストは高くなることがあります。

自動的に実行されるジオコーダーのセットアップ

データが表に追加されるか表の中で更新されるとすぐに、ジオコーダーが、自動的にデータを変換するようにセットアップできます。

ジオコーダーが自動的に実行されるようにセットアップするには、次のことが必要です。
- ジオコーダーの出力からデータを取り込むそれぞれの空間列ごとに、ジオコーディング操作をセットアップする。
- ユーザー ID が、以下の権限または特権をもつ。
  - ジオコーダーを呼び出すトリガーが定義されている表が入ったデータベースに関する、DBADM 権限および DATAACCESS 権限。
  - CONTROL 特権
  - ALTER 特権、SELECT 特権、および UPDATE 特権
  - この表でトリガーを作成するのに必要な特権。

ジオコーダーをバッチ・モードで呼び出す前に、ジオコーダーが自動的に実行されるようにセットアップできます。したがって、自動ジオコーディングは、バッチ・ジオコーディングの前に実行できます。これを行うと、バッチ・ジオコーディングでは、自動的に処理されたデータと同じデータが処理される可能性があります。このような冗長性があっても、データの重複が発生することはありません。これは、空間データが 2 回生成されると、2 番目に生成されたデータが最初のデータをオーバーライドするからです。ただし、パフォーマンスは低下します。

表内の住所データをバッチ・モードでジオコーディングするか、自動モードでジオコーディングするかを決める際には、次のことを考慮に入れてください。

- パフォーマンスは、自動ジオコーディングよりも、バッチ・ジオコーディングのほうがよい。バッチ・セッションは、1 回の初期化で開き、1 回のクリーンアップで終了します。自動ジオコーディングでは、各データ項目は、初期化で始まりクリーンアップで終了する 1 回の操作でジオコーディングされます。
- 全体として、自動ジオコーディングによってデータを読み込む空間列のデータのほうが、バッチ・ジオコーディングによってデータを読み込む空間列のデータよりも、最新のデータである可能性が高い。バッチ・セッションが終わると、次のセッションまで住所データは累積され、ジオコーディングされないままで残っている可能性があります。しかし、自動ジオコーディングが既に使用可能になっている場合は、住所データは、データベースに格納されるとすぐに、ジオコーディングされます。

このタスクは次のように行います。

自動ジオコーディングのセットアップに使用する方法を選択します。

- DB2 コントロール・センターの「ジオコーディングのセットアップ」ウィンドウ、または「ジオコーディング」ウィンドウからセットアップする。
- db2se enable_autogc コマンドを発行する。
- db2gse.ST_enable_autogeocoding ストアード・プロシージャーを呼び出すアプリケーションを実行する。

ジオコーダーをバッチ・モードで実行する

ジオコーダーをバッチ・モードで実行すると、複数のレコードが空間データに変換され、特定の列に入ります。

ジオコーダーをバッチ・モードで実行するには、ユーザー ID が次のいずれかの権限または特権をもっている必要があります。

- データがジオコーディングされる表が入ったデータベースに関する DATAACCESS 権限
- この表に対する CONTROL 特権
- この表に対する UPDATE 特権

毎回のコミットの前に処理するレコードの数を指定するためには、この表に関する SELECT 特権も必要です。ジオコーダーが操作する行を限定するために WHERE 節を指定するときは、これらの節で参照するすべての表やビューに関する SELECT 特権も必要な場合があります。これに関しては、データベース管理者にお尋ねください。

特定の空間列にデータを入れるジオコーダーを実行する前に、その列についてのジオコーディング操作をセットアップできます。操作のセットアップには、ジオコーダーを実行するときに満たされるべき特定の要件を指定することが含まれます。例えば、ジオコーダーが、100 の入力レコードを処理するたびに、DB2 Spatial Extender がコミットを発行するように要求するとします。この場合は、操作をセットアップするときに、要件の数として 100 を指定することになります。

ジオコーダーを実行する準備が整っているときに、操作のセットアップ時に指定した値のうちの任意の値をオーバーライドできます。オーバーライドした値はその実行の間でだけ有効です。

操作をセットアップしない場合は、ジオコーダーを実行するたびに、その実行で満たされるべき要件を指定する必要があります。

このタスクは次のように行います。

バッチ・モードで実行されるジオコーダーの呼び出し方法を選択します。

- DB2 コントロール・センターの「ジオコーディングの実行」ウィンドウからジオコーダーを呼び出す。
- db2se run_gc コマンドを発行する。
- db2gse.ST_run_geocoding ストアード・プロシージャーを呼び出すアプリケーションを実行する。

第 11 章 索引およびビューを使用した空間データへのアクセス

空間列を照会する前に、これにアクセスするための索引およびビューを作成することができます。この章では、以下の項目を説明しています。

- 空間データへのアクセスを迅速に行うために、Spatial Extender が使用する索引の性質
- このような索引の作成方法
- 空間データにアクセスするビューの使用法

空間インデックスのタイプ

データベースの基本表の列について定義された効率よい索引があると、パフォーマンスのよい照会を行うことができます。照会のパフォーマンスは、照会において、その列の値をいかに素早く見つけられるかに直接関係します。索引を使用すれば、照会の実行は速くなり、パフォーマンスは大きく向上します。

空間照会は、通常、複数のディメンションが関係する照会です。例えば、空間照会として、あるポイントがあるエリア (ポリゴン) の中に入っているかどうかを知りたいことがあります。空間照会はマルチディメンションという性質があるため、DB2® ネイティブの B ツリー索引付けは、これらの照会では効率がよくありません。

空間照会は、以下のタイプの索引を使用できます。

- 空間グリッド・インデックス

  DB2 Spatial Extender の索引作成テクノロジーでは、マルチディメンションの空間データの索引を作成するように設計されている*グリッド索引* を利用して、空間列の索引が作成されます。 DB2 Spatial Extender は、地球を平面投影した 2 ディメンション・データ向けに最適化されたグリッド索引を提供しています。
- 測地ボロノイ・インデックス

  DB2 Geodetic Data Management Feature は、複数ディメンションの測地データを含む列についての索引の作成を可能にする、新しい空間アクセス方法をサポートしています。 測地ボロノイ・インデックスは、地球を連続した球体として扱い、極地や、180 度子午線の地点でもゆがみが生じないため、グリッド・インデックスより測地データに適しています。

空間グリッド・インデックス

索引を利用すれば、特に、対象となる表 (複数の場合もある) に含まれる行の数が多い場合に、アプリケーションの照会のパフォーマンス向上に役立ちます。照会オプティマイザーが照会の実行の際に選択するような適切な索引を作成すれば、処理対象となる行数を大幅に減らすことができます。

© Copyright IBM Corp. 1998, 2009

DB2 Spatial Extender は、2 ディメンション・データ向けに最適化されたグリッド索引を提供しています。 索引は形状の X ディメンションと Y ディメンション上に作成されます。

ここでは、グリッド索引についてよく理解できるよう以下の点について説明します。

- 索引の生成
- 照会中での空間処理関数の使用
- 照会による空間グリッド・インデックスの利用の方法

空間グリッド・インデックスの生成

Spatial Extender は、空間グリッド・インデックスを、形状の最小外接長方形 (MBR) を使用して生成します。

ほとんどの形状の場合、MBR はその形状を囲む長方形です。

空間グリッド・インデックスは、領域を、固定サイズ (サイズはユーザーが索引作成時に指定する) の論理的な正方形のグリッドに分割します。グリッド・セルと各形状の MBR の交差部分について 1 つ以上の項目を作成することにより、空間インデックスが空間列に対して作成されます。索引の各項目は、グリッド・セルの ID、形状の MBR、形状を含む行の内部 ID から構成されます。

空間インデックス・レベル (グリッド・レベル) は最大で 3 つ定義できます。複数のグリッド・レベルを使用すると、空間データのいろいろなサイズの索引を最適化できるため、便利です。

4 つ以上のグリッド・セルと交差する形状は、次のより大きなレベルに格上げされます。一般に、大きな形状は、大きなサイズのレベルで索引化されます。形状が、大きなグリッド・サイズでも 10 以上のグリッド・セルと交差する場合には、システム定義のオーバーフロー索引レベルが利用されます。このオーバーフロー・レベルにより、生成される索引項目の数が多くなりすぎるのが防止されます。パフォーマンスを最大限高めるためには、オーバーフロー・レベルの利用を回避できるようなグリッド・サイズを定義する必要があります。

複数のグリッド・レベルが存在する場合、索引作成アルゴリズムは、データを最も細かく索引化するために、可能な限り低いグリッド・レベルを使おうとします。あるレベルの形状が 4 つを超えるグリッド・セルと交差する場合は、次に大きいレベルにプロモートされます (別のレベルがある場合)。したがって、10.0、100.0、そして 1000.0 の 3 つのグリッド・レベルのある空間インデックスは、まず、レベル 10.0 のグリッドと各形状を交差させます。形状が 4 つを超える 10.0 サイズのグリッド・セルと交差する場合、プロモートされてレベル 100.0 のグリッドと交差するようになります。4 つを超える交差が 100.0 レベルで生じていると、この形状は 1000.0 レベルにプロモートされます。10 を超える交差が 1000.0 レベルで生じている場合には、形状はオーバーフロー・レベルで索引化されます。

照会中での空間処理関数の使用

DB2 オプティマイザーは、照会の WHERE 節に以下のいずれかの関数が含まれている場合、空間グリッド・インデックスの利用を検討します。

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

照会による空間グリッド・インデックスの利用の方法

照会オプティマイザーが、空間グリッド・インデックスを選択する際には、照会の実行時に複数ステップのフィルタリング処理が行われます。

フィルタリング処理には以下のステップが含まれます。

1. どのグリッド・セルが照会領域と交差しているかを確認する。*照会領域*とは、処理対象となる形状のことで、ユーザーが、空間処理関数の 2 番目のパラメーターに指定するものです (以下の例を参照)。
2. 索引で、グリッド・セル ID が合致する項目を検索する。
3. 索引項目中の形状の MBR 値を照会領域と比較し、照会領域の外にある値を破棄する。
4. 必要に応じてさらに分析を行う。前のステップまでに得た形状の候補セットに関してさらに分析をし、空間処理関数 (ST_Contains、 ST_Distance など) で一定の条件を満たしているかどうかを確認します。空間処理関数 EnvelopesIntersect を利用すればこのステップをこのステップを省略でき、通常はパフォーマンスを最高にできます。

以下の空間照会の例では、C.GEOMETRY 列についての空間グリッド・インデックスが利用されます。

```
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
```

1 つ目の例では、4 つの座標値によって照会領域が定義されます。これらの座標値は、長方形の左下の隅と右上の隅 (42.0 –73.0 と 43.0 –72.0) を指定するものです。

2 つ目の例では、Spatial Extender はホスト変数に指定された形状 (geometry2) の MBR を算出し、それを照会領域として使用します。

空間グリッド・インデックスを作成する際には、空間アプリケーションが最もよく使用する照会領域のサイズに合うような適切なグリッド・サイズを指定すべきです。グリッド・サイズが大きすぎると、照会領域の外にあるにもかかわらず、照会

領域と交差するグリッド・セル中に存在する、という形状についての索引項目をスキャンすることが増え、この余分なスキャンによってパフォーマンスが低下します。一方、グリッド・サイズが小さすぎると、形状あたりの索引項目が増え、スキャンしなければならない索引項目も増えるため、やはりパフォーマンスが低下します。

DB2 Spatial Extender は、空間列データ分析し、一般的な照会領域のサイズに適合するグリッド・サイズ提案する索引アドバイザー・ユーティリティーを提供しています。

索引レベル数とグリッド・サイズに関する考慮事項

空間グリッド・インデックスの適切なグリッド・サイズを判別するには索引アドバイザーを使用します。これが、索引をチューニングし、空間照会を効率化するための最良の方法です。

グリッド・レベルの数

グリッド・レベルは 3 つまで持つことができます。

空間照会が行われるときは、空間グリッド・インデックス内の各グリッド・レベルについて個別に索引が検索されます。したがって、グリッド・レベルが多ければ多いほど、照会の効率が落ちます。

空間列の値がほぼ同じ相対サイズである場合は、グリッド・レベルを 1 とする。通常、空間列内に同じ相対サイズの形状が複数含まれることはありませんが、空間列中の形状をサイズによってグループ化することは可能です。グリッド・レベルは、この形状グループに対応させる必要があります。

例えば、大きな田舎の土地区画に囲まれた小さな都市区画がグループ化されて入っている空間列を持つ、郡の土地区画の表があるとします。これらの土地区画のサイズを 2 つのグループ (都会の小さな区画と田舎の大きな区画) に分けることができるので、空間グリッド・インデックスも 2 つのグリッド・レベルを指定することになります。

グリッド・セル・サイズ

索引項目数を最少にしながら、グリッド・サイズをできるだけ小さくして最高度の細かさを得るのが、一般的な方法です。

- 列内の小さな形状に対する索引全体を最適化するには、最小グリッド・サイズに小さな値を使用してください。こうすることによって、検索領域内にない形状を測定するオーバーヘッドを省くことができます。ただし、最小グリッド・サイズでは索引項目も最大になります。したがって、照会時に処理される索引項目数が増加すると、索引に必要なストレージ必要量も増加することになります。これらの要因によって、全体のパフォーマンスは低下します。
- 大きなグリッド・サイズを使用すると、大きな形状に対する索引の最適化をさらに進めることができます。大きな形状に大きなグリッド・サイズを使用すると、最小グリッド・サイズで生成されるよりも索引項目は少なくなります。この結果、索引のストレージ要件は低下し、パフォーマンス全体が向上します。

以下に示すいくつかの図を見ると、グリッド・サイズを変えることによる効果がわかります。

図 13 は、土地の区画の図です。個々の区画はポリゴンで表現されています。黒の長方形は、照会領域を表しています。例えば、MBR が照会領域と交差している形状をすべて検索したいとします。図 13 を見ると、28 の形状 (ピンクで強調表示されている) が照会領域と交差する MBR を持っていることがわかります。

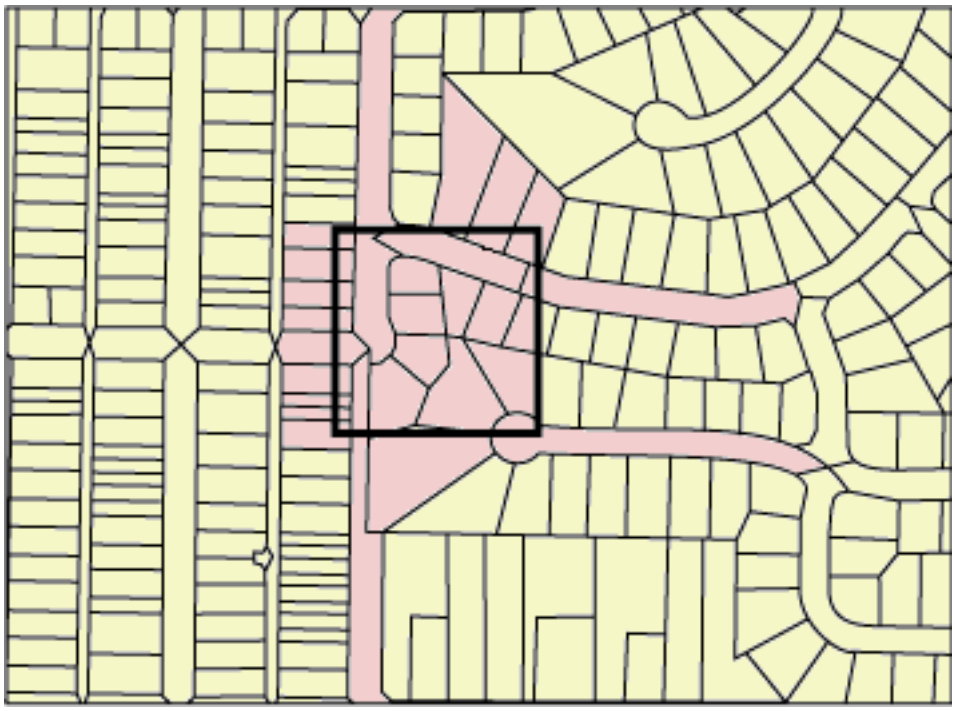


図 13. 近隣の土地の区画

78 ページの図 14 は、グリッド・サイズを照会領域に合わせて小さく (25 に) した例です。

- この場合、照会は、強調表示された 28 の形状のみを戻しますが、 MBR が照会領域と交差している 3 つの余分な形状を調べ、その後で破棄するという処理が必要になります。
- グリッド・サイズを小さくすることで、形状 1 つあたりの索引項目は増えます。照会は実行時に、31 の形状のすべての索引項目にアクセスします。 78 ページの図 14 は、256 のグリッド・セルが、 照会領域に重なっている状態を示しています。この場合、多くの形状が複数の同じグリッド・セルに索引付けされているため、照会を実行すると、578 もの索引項目へのアクセスが必要になります。

この照会領域の場合、グリッド・サイズを小さくすると、余分な索引項目へのスキャンが増えます。

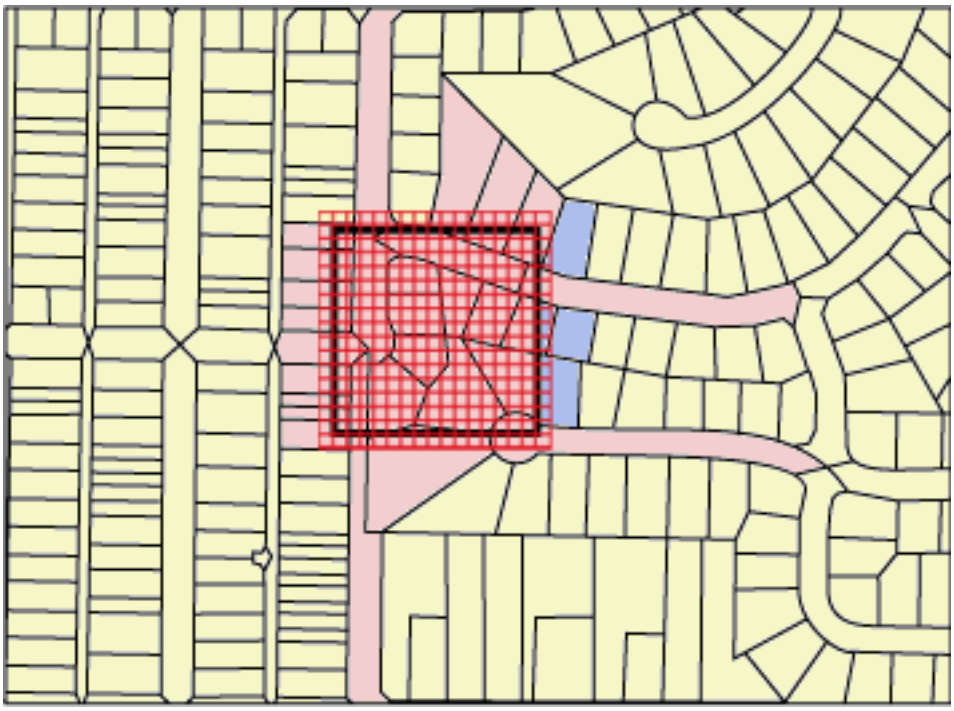


図 14. 土地区画のグリッド・サイズを小さく (25 に) した場合

79 ページの図 15 は、グリッド・サイズを大きく (400®) し、照会領域より広い範囲の形状に重なるようにした例です。

- このようにグリッド・サイズを大きくすると、各形状に対応する索引項目は 1 つのみになりますが、照会は、MBR がグリッド・セルと交差する 59 もの余分な形状について調べ、破棄するという処理を行わねばなりません。
- 実行中、照会は、照会領域と交差する 28 の形状のすべての索引項目にアクセスし、さらに、59 の余分な形状の索引項目にもアクセスするため、合計で 112 の索引項目にアクセスすることになります。

この照会領域の場合、グリッド・サイズを大きくすると、数多くの余分な形状へのアクセスが必要になります。

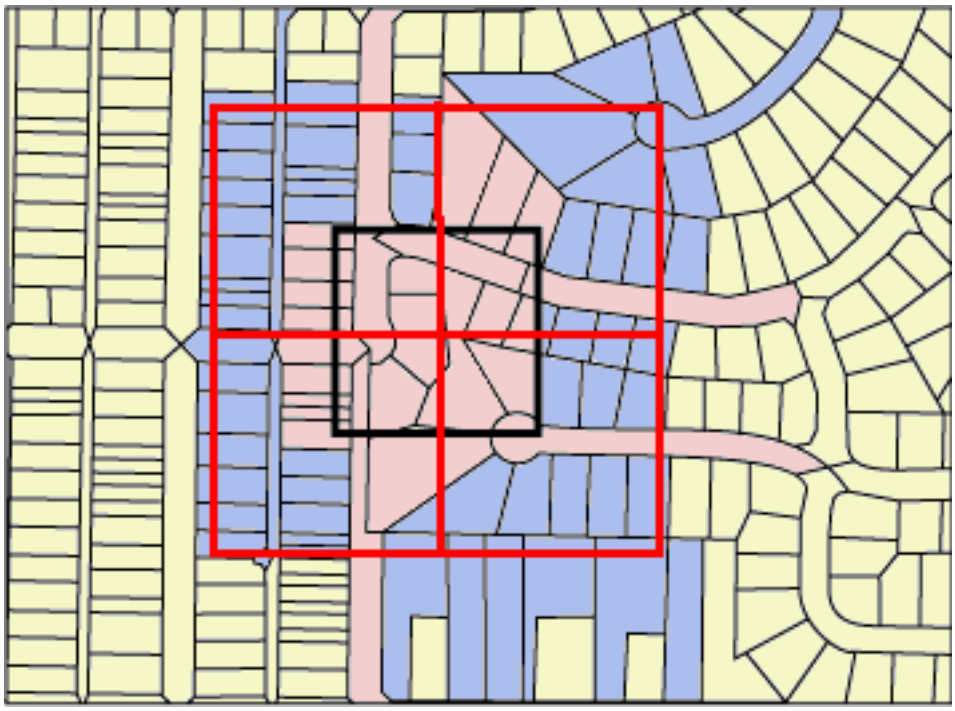


図 15. グリッド・サイズを大きく (400 に) した場合

80 ページの図 16 は、グリッド・サイズを照会領域に合わせ、中程度に (100 に) した例です。

- この場合、照会は、強調表示された 28 の形状のみを戻しますが、MBR が照会領域と交差している 5 つの余分な形状を調べ、その後で破棄するという処理が必要になります。
- 実行中、照会は、照会領域と交差する 28 の形状のすべての索引項目にアクセスし、さらに、5 つの余分な形状の索引項目にもアクセスするため、合計で 91 の索引項目にアクセスすることになります。

この照会領域の場合は、中程度のグリッド・サイズが最適ということになります。グリッド・サイズが小さい場合に比べ、アクセスする索引項目が大きく減り、グリッド・サイズが大きい場合に比べ、余分な形状にアクセスすることも減るからです。

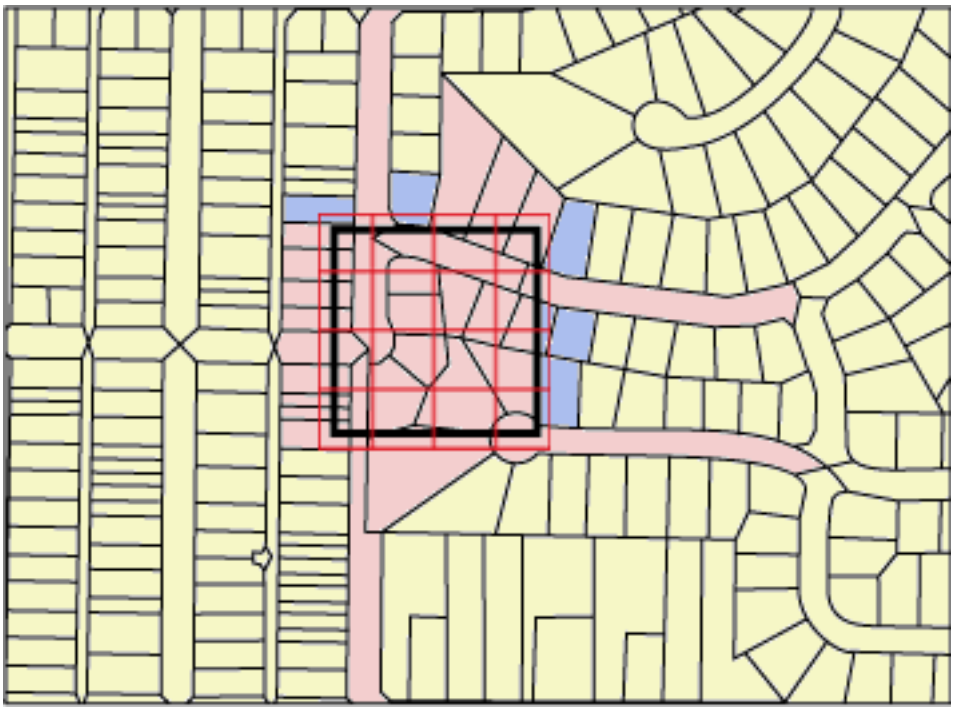


図 16. グリッド・サイズを中程度に (100 に) した場合

空間グリッド索引の作成

空間グリッド・インデックスを作成して、空間照会の最適化に役立つように空間列に 2 ディメンションのグリッド・インデックスを定義します。

空間グリッド・インデックスを作成するには、以下のことが条件になります。

- ユーザー ID に DB2 SQL CREATE INDEX ステートメントに必要な権限を持たせる。ユーザー ID には、次の権限、あるいは特権のうちの少なくとも 1 つが必要です。
  - 列を含む表が入っているデータベースに関する DBADM 権限
  - 以下に示す権限、特権の両方。
    - 以下の表特権のいずれか。
      - 表に対する CONTROL 特権
      - 表に対する INDEX 特権
    - スキーマに対する以下の権限、あるいは特権のうちのいずれか。
      - 索引のスキーマが存在しない場合は、データベースに対する IMPLICIT_SCHEMA 権限
      - 索引のスキーマ名が既存のスキーマを参照する場合は、そのスキーマに対する CREATEIN 特権
- 完全修飾の空間グリッド・インデックス名に指定する値と、索引が使用する 3 つのグリッドのサイズが分かっている。

推奨事項:

- 列に関する空間グリッド索引を作成するときは、その前に索引アドバイザーを使用して、索引のパラメーターを決めておきます。索引アドバイザーは、空間列データを分析し、その空間グリッド索引に適切なグリッド・サイズを提案できます。
- データを列に初期ロードしようとする場合は、ロード・プロセスが完了した後で空間グリッド・インデックスを作成する必要があります。 こうすることによって、索引アドバイザーを使用して得られたデータ特性に基づく最適のグリッド・セル・サイズを選択できます。 さらに、索引の作成前にデータをロードすることによって、ロード・プロセスのパフォーマンスが向上します。これは、ロード・プロセス中に空間グリッド・インデックスを保守する必要がないためです。

制約事項:
空間グリッド・インデックスを作成するときも、CREATE INDEX ステートメントを使用して索引を作成する際の制約が当てはまります。すなわち、索引を作成する列は、ビュー列またはニックネーム列ではなく、基本表の列でなければなりません。DB2 データベース・システムは、処理中に別名を解決します。

空間グリッド・インデックスは、空間列に対する照会のパフォーマンスを向上するために作成します。

作成する空間グリッド・インデックスには、以下のような情報を持たせます。

- 名前
- 空間グリッド・インデックスが定義される空間列の名前
- 3 つのグリッド・サイズを組み合わせることで、索引項目数の合計と、照会に対応するためにスキャンしなければならない索引項目数を最小にでき、パフォーマンスを最適化することができます。

空間グリッド・インデックスは、以下のいずれかの方法によって作成できます。

- DB2 コントロール・センターの「Spatial Extender」ウィンドウを使用する。
- EXTEND USING 節で db2gse.spatial_index 拡張子を付けて SQL CREATE INDEX ステートメントを使用する。
- DB2 Spatial Extender で働く GIS ツールを使用する。索引を作成するためにこのようなツールを使用すると、前述したのと同じ SQL CREATE INDEX ステートメントがツールから出されます。

このトピックでは、最初の 2 つの方法のためのステップを示します。空間グリッド索引を作成するための GIS ツールの使用についての詳細は、このツールに付属する資料を参照してください。

このタスクは次のように行います。

SQL CREATE INDEX を使用した空間グリッド・インデックスの作成

1. EXTEND USING 節と db2gse.spatial_index グリッド・インデックス拡張子を使用して CREATE INDEX ステートメントを決定する。 例えば、以下のステートメントでは、TERRITORY 空間列を持つ BRANCHES 表に対応する TERRIDX 空間グリッド・インデックスが作成されます。

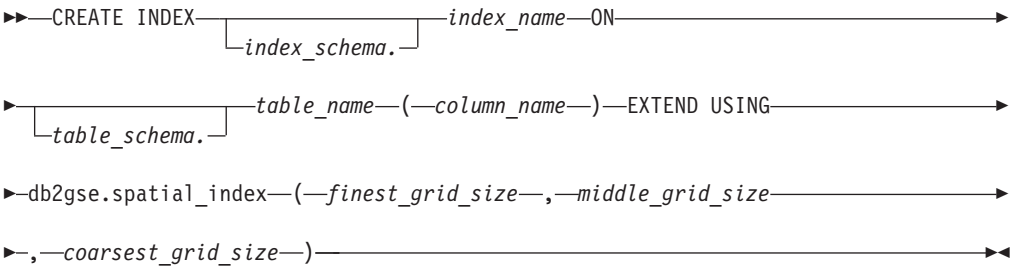
```
CREATE INDEX terridx
   ON branches (territory)
   EXTEND USING db2gse.spatial_index (1.0, 10.0, 100.0)
```

2. DB2 コマンド・エディター、 DB2 コマンド・ウィンドウ、DB2 コマンド行プロセッサーで CREATE INDEX コマンドを発行する。

空間グリッド・インデックスを作成するための CREATE INDEX ステートメント

空間グリッド・インデックスを作成する場合は、CREATE INDEX ステートメントに EXTEND USING 節を指定して使用してください。

構文



パラメーター

index_schema.
作成する索引が属するスキーマの名前。名前を指定しないと、DB2 データベース・システムは CURRENT SCHEMA 特殊レジスターに保管されているスキーマ名を使用します。

index_name
作成するグリッド索引の非修飾名。

table_schema.
column_name を含んでいる表が属するスキーマの名前。名前を指定しないと、DB2 は CURRENT SCHEMA 特殊レジスターに保管されているスキーマ名を使用します。

table_name
column_name を含んでいる表の非修飾名。

column_name
空間グリッド・インデックスを作成する空間列の名前。

finest_grid_size、middle_grid_size、coarsest_grid_size
空間グリッド・インデックスのためのグリッド・サイズ。これらのパラメーターは、以下の条件を満たしている必要があります。

- *finest_grid_size* は 0 より大きくなければならない。
- *middle_grid_size* は *finest_grid_size* より大きいか、0 でなければならない。
- *coarsest_grid_size* は *middle_grid_size* より大きいか、0 でなければならない。

空間グリッド・インデックスを作成するときは、コントロール・センターを使用する場合も、CREATE INDEX ステートメントを作成する場合も、最初の形状の索引作成時にグリッド・サイズの有効性がチェックされます。したがって、指定したグリッド・サイズの値が以上の条件に合致していないと、以下に説明するような状況でエラー状態が発生します。

- 空間列のすべて形状が NULL であれば、Spatial Extender はグリッド・サイズの妥当性を確認することなく、索引を作成することができます。 Spatial Extender は、空間列に NULL でない形状が挿入されるか、空間列中の NULL でない形状が更新される場合に、グリッド・サイズの妥当性を確認します。指定されたグリッド・サイズが無効である場合は、非 NULL の形状が挿入または更新された時点でエラーが発生します。
- 索引作成時に空間列に非 NULL の形状が存在した場合、Spatial Extender はその時点でグリッド・サイズの妥当性を確認します。指定されたグリッド・サイズが無効である場合は、すぐにエラーが発生し、空間グリッド・インデックスは作成されません。

例

以下の例では、CREATE INDEX ステートメントは、BRANCHES 表の TERRITORY 空間列に TERRIDX 空間グリッド・インデックスを作成します。

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

索引アドバイザーを使用した空間グリッド・インデックスのチューニング

索引アドバイザーを使用した空間グリッド・インデックスのチューニング—概要

DB2® Spatial Extender には、索引アドバイザー と呼ばれるユーティリティーが準備されています。これを使用すると、以下のことが行えます。

- 空間グリッド・インデックスに適したグリッド・サイズを判別する。

  索引アドバイザーは、空間列中の形状を分析し、空間グリッド・インデックスに最適なグリッド・サイズを推奨します。
- 既存のグリッド索引を分析する。

  索引アドバイザーは、現在のグリッド・セル・サイズが空間データの検索にどれほど役立っているかを判断するのに利用できる統計情報を収集、表示することができます。

空間グリッド・インデックス作成のためのグリッド・サイズの決定

前提条件

索引作成対象のデータを分析するには、以下の条件が満たされている必要があります。

- ユーザー ID にこの表に対する SELECT 特権があること。

- 表の行数が 100 万を超える場合には、処理時間を適正に保つために、ANALYZE 節を使用して行のサブセットを分析しなければならない場合もあります。ANALYZE 節を使用できる USER TEMPORARY 表スペースが必要です。この表スペースのページ・サイズは最低 8 KB に設定する必要があり、ユーザーには、表スペースに対する USE 特権が必要です。例えば以下の DDL ステートメントは、ユーザー TEMPORARY 表スペースと同じページ・サイズのバッファー・プールを作成し、ユーザーに USE 特権を付与します。

  ```
  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;
  ```

  あるいは DB2 コントロール・センターを使用し、ユーザー表スペースと、それに対応するバッファー・プールを作成することもできます。

列に空間グリッド・インデックスを作成するときは、その前に索引アドバイザーを使用して、適切なグリッド・サイズを判別できます。

このタスクは次のように行います。

空間グリッド・インデックス作成のための適切なグリッド・サイズは以下の手順で決定します。

1. 作成する索引に対する推奨グリッド・セル・サイズを、索引アドバイザーを使って求めます。
   a. 索引アドバイザーを呼び出すコマンドを次のように ADVISE キーワードを指定して入力し、グリッド・セル・サイズを要求します。例えば、索引アドバイザーを、COUNTIES 表の SHAPE 列に対して実行するには、以下のように入力します。

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

      制約事項: 上記の gseidx コマンドをオペレーティング・システムのプロンプトから入力する場合は、コマンド全体を単一行で入力する必要があります。もう 1 つの方法として、 CLP ファイルから gseidx コマンドを実行することができます。この方法では、コマンドを複数の行に分割することができます。

      索引アドバイザーは、推奨グリッド・セル・サイズを戻します。 ADVISE キーワードを指定した上記の gseidx コマンドでは、例えば次のような SHAPE 列の推奨セル・サイズが戻されます。

      ```
      Query Window Size  Suggested Grid Sizes             Cost
      -----------------  -------------------------       ----
             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. gseidx の出力から、画面に表示する座標の幅を基に適切な照会領域サイズを選択します。

この例では、10 進数の緯度と経度の値が座標を表しています。例えば、地図の画面の幅が、通常、約 0.5 度 (約 55 km に相当する) である場合には、Query Window Size 列の値が 0.5 になっている行を選択します。この行のグリッド・サイズは、 1.4、3.5、14.0 となっています。

2. 推奨されたグリッド・サイズで索引を作成します。上記の例の場合、以下のような SQL ステートメントを実行することになります。

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

空間グリッド・インデックスに関する統計の分析

前提条件

索引作成対象のデータを分析するには、以下の条件が満たされている必要があります。

- ユーザー ID にこの表に対する SELECT 特権があること。
- 表の行数が 100 万を超える場合には、処理時間を適正に保つために、ANALYZE 節を使用して行のサブセットを分析しなければならない場合もあります。
  ANALYZE 節を使用できる USER TEMPORARY 表スペースが必要です。この表スペースのページ・サイズは最低 8 KB に設定する必要があり、ユーザーには、表スペースに対する USE 特権が必要です。例えば、以下の DDL ステートメントは、ユーザー TEMPORARY 表スペースと同じページ・サイズのバッファー・プールを作成し、ユーザーに USE 特権を付与します。

```
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;
```

  あるいは DB2 コントロール・センターを使用し、ユーザー表スペースと、それに対応するバッファー・プールを作成することもできます。

既存の空間グリッド索引に関する統計を見れば、そのグリッド索引が効率のよいものであるか、あるいは、もっと効率のよい索引に置き換える必要があるかが分かります。統計を入手する場合、および必要に応じて索引を置き換えるには、索引アドバイザーを使用します。

ヒント: 索引が照会によって使用されているかを確認することは、索引のチューニングと同じくらい重要です。空間インデックスが使用されているかどうかを確認するには、DB2 コントロール・センターで Visual Explain を実行するか、db2exfmt などのコマンド行ツールを照会に対して使用します。実行結果の『Access Plan (アクセス・プラン)』 セクションに、EISCAN 演算子や、該当する空間インデックスの名前がある場合には、照会がその索引を使用していることを意味します。

このタスクは次のように行います。

空間グリッド・インデックスに関する統計を入手する手順、および必要に応じて索引を置き換える手順は以下のとおりです。

1. 既存の索引のグリッド・セル・サイズに基づく統計を索引アドバイザーに集めさせる。統計は、索引が付けられたデータのサブセット、あるいはすべてのデータについて要求できます。
   - 行のサブセット内の索引が付いたデータに関する統計を入手するには、gseidx コマンドを入力し、ANALYZE キーワードとそのパラメーターを、existing-index 節と DETAIL キーワードに加えて指定します。統計を得るために索引アドバイザーが分析する行の数またはパーセンテージを指定できます。例えば、COUNTIES_SHAPE_IDX という索引の付いたデータのサブセットについての統計を得るには、以下のようなコマンドを実行します。

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

   - 索引が付いたすべてのデータに関する統計を入手するには、gseidx コマンドを入力し、existing-index 節を指定する。 DETAIL キーワードを含めます。例えば、索引アドバイザーを COUNTIES_SHAPE_IDX という索引に対して実行するには、以下のように入力します。

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

   索引アドバイザーは、既存の索引の統計、データのヒストグラム、推奨セル・サイズを戻します。例えば COUNTIES_SHAPE_IDX の索引が付いたすべてのデータについての上記 gseidx コマンドは、次のような統計を戻します。

```
Grid Level 1
------------

Grid Size                     : 0.5
Number of Geometries          : 2936
Number of Index Entries       : 12197

Number of occupied Grid Cells : 2922
Index Entry/Geometry ratio    : 4.154292
Geometry/Grid Cell ratio      : 1.004791
Maximum number of Geometries per Grid Cell: 14
Minimum number of Geometries per Grid Cell: 1

Index Entries :  1      2      3      4      10
---------------  ------ ------ ------ ------ ------
Absolute      :  86     564    72     1519   695
Percentage (%):  2.93   19.21  2.45   51.74  23.67



Grid Level 2
------------

Grid Size                     : 0.0
No geometries indexed on this level.


Grid Level 3
------------

Grid Size                     : 0.0
No geometries indexed on this level.
```

```
Grid Level X
------------

Number of Geometries          : 205
Number of Index Entries       : 205
```

2. 既存の索引のグリッド・セル・サイズが、検索でどの程度うまく機能しているかを判断する。前のステップで戻された統計を確認する。

   ヒント:

   - 統計『Index Entry/Geometry ratio』は、 1 から 4 の範囲、できれば、1 に近い値が望ましいと言えます。
   - 形状ごとの索引項目の数がオーバーフロー・レベルに達しないよう、最高でも 10 以下になるよう、グリッド・サイズを調整します。

     索引アドバイザーの出力に 『Grid Level X』セクションが表示された時は、オーバーフロー・レベルの索引エントリーが存在することを意味します。

   前のステップで COUNTIES_SHAPE_IDX について得られたインデックス統計を見ると、以下のような理由から、グリッド・サイズ (0.5, 0, 0) がこの列のデータに対して適切でないことがわかります。

   - Grid Level 1 で、『Index Entry/Geometry ratio』 の値 4.154292 が、ガイドラインである 4 を上回っている。

     『Index Entries』 行の 1、2、3、4、10 という値は、形状ごとの索引項目の数を示しています。 『Index Entries』 列の下の 『Absolute』 値は、特定の数の索引項目を持つ形状の数を示しています。例えば、前のステップの出力を見ると、1519 の形状が 4 つの索引項目を持っていることが分かります。索引項目数 10 に対応する 『Absolute』 値は 695 なので、695 の形状が 5 から 10 の間の索引項目を持っていることになります。
   - 『Grid Level X』セクションが表示された時は、オーバーフロー・レベルの索引エントリーが存在することを意味します。ここでは、205 の形状が 10 を超える索引項目を持っていることが示されています。

3. 統計が十分なものでなければ、索引アドバイザーの出力中の、Histogram セクションで、Query Window Size、Suggested Grid Sizes 列の適切な行を確認します。
   a. 形状の数が最高になる MBR サイズを探します。『Histogram』 セクションには、MBR サイズと形状数の対応表が表示されます。以下の例では、最高の形状数 (437) は MBR サイズ 0.5 に対応しています。

```
Histogram:
----------
    MBR Size             Geometry Count
    -------------------- --------------------
                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. Query Window Size 行から、値 0.5 を探すと、推奨グリッド・サイズが (1.4, 3.5, 14.0) であるとわかります。

```
Query Window Size  Suggested Grid Sizes             Cost
-----------------  --------------------------      ----
              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. この推奨グリッド・サイズが、ステップ 2 のガイドラインに合うか確認します。そのために、推奨グリッド・サイズを指定して gseidx コマンドを実行します。

```
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)

Grid Level 1
------------

Grid Size                     : 1.4
Number of Geometries          : 3065
Number of Index Entries       : 5951

Number of occupied Grid Cells : 513
Index Entry/Geometry ratio    : 1.941599
Geometry/Grid Cell ratio      : 5.974659
Maximum number of Geometries per Grid Cell: 42
Minimum number of Geometries per Grid Cell: 1
```

```
Index Entries :  1      2      3      4      10
---------------  ------ ------ ------ ------ ------
Absolute      :  1180   1377   15     493    0
Percentage (%):  38.50  44.93  0.49   16.08  0.00


Grid Level 2
------------

Grid Size                      : 3.5
Number of Geometries           : 61
Number of Index Entries        : 143

Number of occupied Grid Cells : 56
Index Entry/Geometry ratio    : 2.344262
Geometry/Grid Cell ratio      : 1.089286
Maximum number of Geometries per Grid Cell: 10
Minimum number of Geometries per Grid Cell: 1

Index Entries :  1      2      3      4      10
---------------  ------ ------ ------ ------ ------
Absolute      :  15     28     0      18     0
Percentage (%):  24.59  45.90  0.00   29.51  0.00


Grid Level 3
------------

Grid Size                      : 14.0
Number of Geometries           : 15
Number of Index Entries        : 28

Number of occupied Grid Cells : 9
Index Entry/Geometry ratio    : 1.866667
Geometry/Grid Cell ratio      : 1.666667
Maximum number of Geometries per Grid Cell: 10
Minimum number of Geometries per Grid Cell: 1

Index Entries :  1      2      3      4      10
---------------  ------ ------ ------ ------ ------
Absolute      :  7      5      1      2      0
Percentage (%):  46.67  33.33  6.67   13.33  0.00
```

この統計を見ると、推奨グリッド・サイズはガイドラインに合っていることがわかります。

- 『Index Entry/Geometry ratio』 の値は、Grid Level 1 で 1.941599、Grid Level 2 で 2.344262、Grid Level 3 で 1.866667 です。どの値も、1 から 4 までの間というガイドラインに合致しています。
- 『Grid Level X』 セクションが表示されなければ、オーバーフロー・レベルの索引項目が存在しないことを意味します。

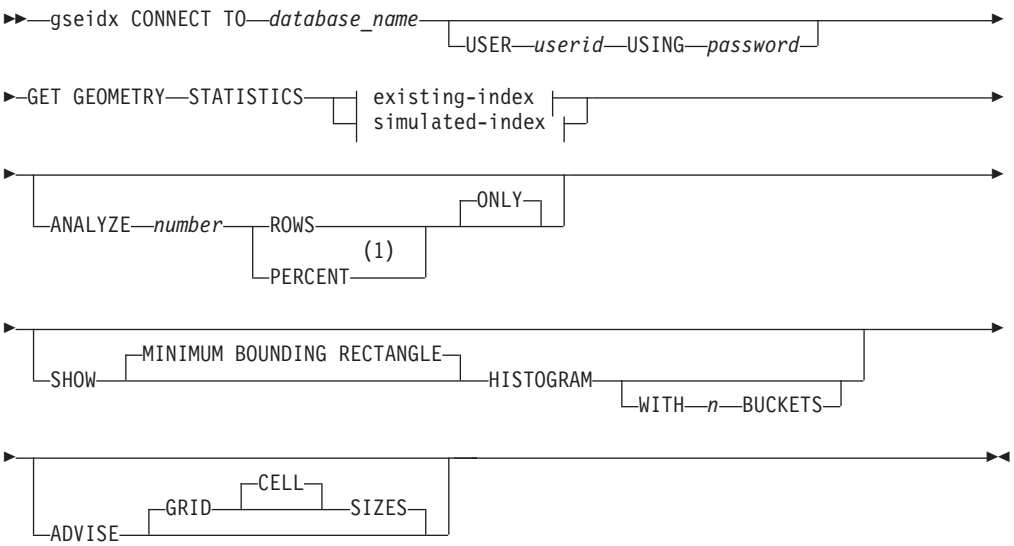
5. 既存の索引をドロップして、推奨されたグリッド・サイズに合う索引に置き換える。ここで使用した例の場合であれば、以下の DDL ステートメントを実行することになります。

```
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);
```

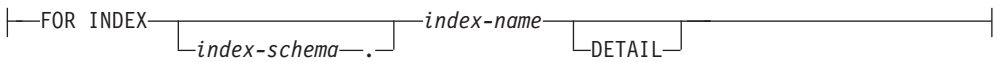
gseidx コマンド

gseidx コマンドは、索引アドバイザーを空間グリッド・インデックスに対して実行する際に使用します。

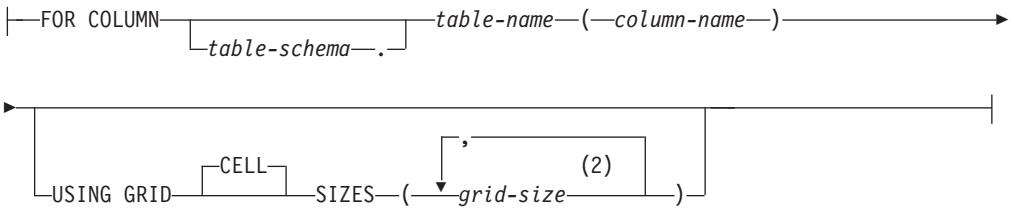
構文



existing-index:



simulated-index:



注:

1 PERCENT キーワードの代わりに、パーセント記号 (%) を指定できます。

2 1、2、または 3 つのグリッド・レベルでセル・サイズを指定できます。

パラメーター

database_name
空間表が置かれているデータベースの名前

userid 索引または表が置かれているデータベースに対する DATAACCESS 権限、あるいは表に対する SELECT 権限をもつユーザー ID。データベース所有者のユーザー ID で DB2 コマンド環境にログオンした場合は、gseidx コマンドで *userid*、*password* を指定する必要はありません。

password
ユーザー ID のパスワード。

existing-index

どの既存索引について統計を集めるかを指定する既存索引名。

index-schema
既存の索引を含んでいるスキーマの名前。

index-name
既存索引の非修飾名。

DETAIL
各グリッド・レベルについて以下の情報を示します。
- グリッド・セルのサイズ
- 索引が付いた形状の数
- 索引項目数
- 形状を含んでいるグリッド・セルの数
- 形状当たりの平均索引項目数
- グリッド・セル当たりの平均形状数
- 最も多く形状を含んでいるセル内の形状の数
- 含まれている形状が最も少ないセル内の形状の数

simulated-index

表の列と、その列用にシミュレートされた索引を指します。

table-schema
シミュレートされた索引の対象となっている列を持つ表が入っているスキーマの名前

table-name
シミュレート索引の対象となっている列を持つ表の非修飾名。

column-name
シミュレート索引の対象となっている表列の非修飾名。

grid-size
シミュレートされた索引の各グリッド・レベル (最も密、中程度、最も粗い) のセルのサイズ。少なくとも 1 つのレベルのセル・サイズを指定する必要があります。レベルを含めたくない場合は、レベルに対してグリッド・セル・サイズを指定しないか、またはレベルにグリッド・セル・サイズとして 0 を指定します。

grid-size パラメーターを指定すると、索引アドバイザーは、existing-index 節に DETAIL キーワードが指定されている場合と同じ種類の統計を戻します。

ANALYZE number ROWS | PERCENT ONLY
表の行のサブセット内のデータに関する統計を集める。表の行数が 100 万を超

える場合には、処理時間を適正に保つために ANALYZE 節を使用しなければならない場合もあります。このサブセットに含めるおよその行数またはおよそのパーセンテージを指定します。

SHOW MINIMUM BOUNDING RECTANGLE HISTOGRAM
形状の最小外接長方形 (MBR) のサイズ、および MBR が同じサイズである形状の数を示すグラフを表示させます。

WITH n BUCKETS
分析されたすべての形状の MBR に関するグループ化の数を指定する。小さい MBR はその他の小さい形状とまとめてグループ化されます。大きい MBR は、他の大きい形状とともにグループ化されます。

このパラメーターを指定しなかった場合、あるいは 0 を指定した場合には、索引アドバイザーは対数のバケット・サイズを表示します。MBR サイズは、例えば 1.0、2.0、3.0... 10.0、20.0、30.0...100.0、 200.0、300.0 といった対数値になります。

0 より大きいバケット数を指定すると、索引アドバイザーは、等サイズの値を表示します。例えば、MBR サイズは 8.0、16.0、24.0...320.0、328.0、334.0 といった等サイズの値になります。

デフォルトでは、対数サイズのバケットが使用されます。

ADVISE GRID CELL SIZES
最適化サイズに近いグリッド・セル・サイズを計算します。

使用上の注意

gseidx コマンドをオペレーティング・システムのプロンプトから入力する場合は、コマンド全体を単一行で入力する必要があります。

例

以下の例は、名前が COUNTIES_SHAPE_IDX である既存のグリッド索引に関する詳細情報を戻し、適切なグリッド索引サイズを提示するための要求です。

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

ビューを使用した空間列へのアクセス

他のデータ・タイプ用に DB2 にビューを定義するのと同じ方法で、空間列を使用するビューを定義できます。

空間列を持つ表があり、ビューでそれを使用した場合は、以下の情報ソースを参照してください。

第 12 章 空間情報の分析および生成

空間列にデータを入れると、それらを照会できるようになります。この章では、以下の項目を説明しています。

- 照会をサブミットできる環境
- 照会の中から呼び出すことができる各種タイプの空間処理関数の例
- 空間処理関数を空間インデックスと組み合わせて使用する場合のガイドライン

空間分析を行うための環境

以下のプログラミング環境で、SQL および空間処理関数を使用して、空間分析を行うことができます。

- 対話式 SQL ステートメント。

  対話式 SQL ステートメントは、DB2 コマンド・エディター、DB2® コマンド・ウィンドウ、または DB2 コマンド行プロセッサーから入力できます。
- DB2 がサポートするすべての言語で書かれたアプリケーション・プログラム。

空間処理関数による操作の例

DB2 Spatial Extender には、空間データに関する各種の操作を行う関数が準備されています。一般的に、これらの関数は、実行する操作のタイプに従って分類できます。表 3 は、それらのカテゴリーを例とともに示したものです。表 3 に続けて、これらの例のためのコーディングが示してあります。

表 3. 空間処理関数および操作

関数のカテゴリー	操作の例
特定の形状についての情報を戻す。	Store 10 の販売区域の範囲を平方マイルで戻す。
比較を行う。	顧客の家が Store 10 の販売区域内にあるかどうかを判断する。
既存の形状から新しい形状を派生させる。	そのロケーションから店の販売地域を導き出す。
形状をデータ交換フォーマットとの間で変換する。	GML フォーマットの顧客情報を形状に変換し、その情報を DB2 データベースに加えられるようにする。

例 1: 特定の形状についての情報を戻す

この例では、ST_Area 関数が Store 10 の販売地域を表す数値を戻します。この関数は、この区域のロケーションを定義するために使用される座標システムと同じ単位でこの区域を戻します。

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

© Copyright IBM Corp. 1998, 2009

次の例は、前の例と同じ操作を示していますが、今度は、ST_Area をメソッドとして呼び出し、平方マイルの単位でこの区域を戻します。

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

例 2: 比較を行う

この例では、ST_Within 関数が、顧客の家を表す形状の座標を、Store 10 の販売地域を表す形状の座標と比較します。この関数の出力によって、居住域が販売区域内にあるかどうかが分かります。

```
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
```

例 3: 既存の形状から新しい形状を派生させる

この例では、関数 ST_Buffer によって、商店のロケーションを表す形状から、商店の販売地域を表す形状が引き出されます。

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

次の例は、前の例と同じことをしますが、今度は ST_Buffer をメソッドとして呼び出しています。

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

例 4: データ交換フォーマットと形状とを互いに変換する

この例では、GML でコーディングされている顧客情報を形状に変換して、その情報を DB2 データベースに保管できるようにします。

```
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) )
```

照会を最適化するために索引を使用する関数

*比較関数*と呼ばれるいくつかの特殊な空間処理関数は、空間グリッド・インデックス、あるいは測地ボロノイ・インデックス (いずれも *空間インデックス*とも呼ばれる) を利用することで照会のパフォーマンスを向上できます。これらの関数は、それぞれ、2 つの形状の比較を行います。比較結果が一定の基準に合致していれば、関数は値 1 を戻します。結果が基準に合わない場合は、関数は値 0 を戻し ます。比較が行えない場合、関数は値 NULL を戻すことがあります。

例えば、関数 ST_Overlaps は、同じディメンション (例えば、2 つの折れ線または 2 つのポリゴン) をもつ 2 つの形状を比較します。2 つの形状が部分的にオーバーラップし、かつ、オーバーラップするスペースがそれらの形状と同じディメンションを持つ場合は、ST_Overlaps は値 1 を戻します。

表 4 は、個々の比較関数が、空間グリッド・インデックス、測地ボロノイ・インデックスを使用できるかどうかをまとめたものです。

表 4. 比較関数による空間グリッド・インデックス、測地ボロノイ・インデックスの使用の可否

比較関数	空間グリッド・インデックスを使用できるか	測地ボロノイ・インデックスを使用できるか
EnvelopesIntersect	はい	はい
ST_Contains	はい	はい
ST_Crosses	はい	いいえ
ST_Distance	はい	はい
ST_EnvIntersects	はい	はい
ST_Equals	はい	いいえ
ST_Intersects	はい	はい
ST_MBRIntersects	はい	はい
ST_Overlaps	はい	いいえ
ST_Touches	はい	いいえ
ST_Within	はい	はい

関数を実行するのに必要な時間とメモリーが原因で、このような実行には、かなりの処理を必要とする場合があります。さらに、比較する形状が複雑であればあるほど、比較は複雑になり、時間がかかります。上に列挙した比較関数は、形状の位置を探すのに空間インデックスを使用できれば、その操作をより速く完了できます。このような関数で空間インデックスを使用できるようにするには、以下のすべての規則を守ってください。

- 関数は WHERE 節に指定する必要がある。SELECT、HAVING または GROUP BY 節に指定した場合は、空間インデックスは使用できません。
- 関数は、述部の左側の式でなければならない。
- 関数の結果を他の式と比較する述部で使用する演算子は、唯一の例外として ST_Distance 関数で小なり演算子を使用するのを除き、等号である必要がある。
- 述部の右側の式は、ST_Distance 関数が左側にある場合を除き、定数1 でなければならない。
- 操作は、空間インデックスが定義されている空間列の検索が絡むものでなければならない。

例えば、以下のように指定します。

```
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
```

96 ページの表 5 は、空間インデックスを活用する空間照会の正しい作成方法と間違った作成方法を示しています。

表 5. 空間処理関数が空間インデックスを活用するための規則に準拠している例と違反している例

空間処理関数を参照する照会	違反の内容
SELECT * FROM stores AS s WHERE db2gse.ST_Contains(s.sales_zone, ST_Point(-121.8,37.3, 1)) = 1	この例は、どの条件にも違反していない。
SELECT * FROM stores AS s WHERE db2gse.ST_Length(s.location) > 10	空間処理関数 ST_Length は、形状を比較せず、空間インデックスを使用できない。
SELECT * FROM stores AS s WHERE 1=db2gse.ST_Within(s.location,:BayArea)	関数は、述部の左側の式でなければならない。
SELECT * FROM stores AS s WHERE db2gse.ST_Contains(s.sales_zone, ST_Point(-121.8,37.3, 1)) <> 0	等しいかどうかの比較には、整数の定数 1 を使用する必要がある。
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	この関数のどちらの引数にも空間インデックス が存在しない。したがって、索引は使用できない。

第 13 章 DB2 Spatial Extender コマンド

この章では、DB2 Spatial Extender のセットアップに使用されるコマンドを説明しています。 また、これらのコマンドを使用してプロジェクトを開発する方法も説明しています。

DB2 Spatial Extender のセットアップとプロジェクト開発用のコマンドの呼び出し

db2se と呼ばれるコマンド行プロセッサー (CLP) を使用して、Spatial Extender をセットアップし、空間データを使用するプロジェクトを作成します。このトピックでは、db2se を使用して DB2 Spatial Extender コマンドを実行する方法を説明します。

前提条件

db2se コマンドを出すには、そのための権限が必要です。特定のコマンドに必要な権限については、そのコマンドの関連ストアード・プロシージャーが記載してある 98 ページの表 6 を参照してください。例えば、db2se create_srs コマンドには、db2.ST_create_srs ストアード・プロシージャーと同じ権限が必要です。

例外: db2se shape_info コマンドは、ストアード・プロシージャーを呼び出しません。その代わり、形状ファイルの説明に関する情報を表示します。

オペレーティング・システムのプロンプトから db2se コマンドを入力します。

指定できるサブコマンドやパラメーターを調べるには、以下のようにします。

- db2se または db2se -h を入力してから、Enter キーを押す。 db2se サブコマンドのリストが表示されます。
- db2se とサブコマンドを入力する、あるいは db2se とサブコマンドに続けて -h を入力する。次に Enter キーを押します。サブコマンドに必要な構文が表示されます。この構文には、以下のような規則があります。
  - 各パラメーターの前にはダッシュが付き、後にはパラメーター値のプレースホルダーが続く。
  - 大括弧で囲まれたパラメーターはオプションである。その他のパラメーターは必須である。

重要: ユーザーの便宜のためにコマンド構文をモニターに対話式に表示できるので、別の場所で構文を調べる必要はありません。

db2se コマンドを出すには、`db2se` と入力します。次に、サブコマンドを入力し、後に、サブコマンドに必要なパラメーターとパラメーター値を続けて入力します。最後に、Enter キーを押します。

指定したばかりのデータベースにアクセスできるようにするために、ユーザー ID とパスワードの入力が必要になる場合があります。例えば、自分とは異なるユーザ

© Copyright IBM Corp. 1998, 2009

ーとして、データベースに接続しようとする場合は、ID とパスワードを入力します。常に、ID の前にはパラメーターの userId を、パスワードの前にはパラメーターの pw を付けます。

ユーザー ID とパスワードを指定しない場合は、現行のユーザー ID とパスワードがデフォルトとして使用されます。入力する値は、デフォルトでは大文字小文字の区別がありません。大文字小文字の区別をしたい場合は、値を二重引用符で囲みます。たとえば、小文字の表名 mytable を指定するには、 ″mytable″ と入力します。

引用符がシステム・プロンプト (シェル) で解釈されないようにするため、引用符をエスケープする必要がある場合があります。例えば、¥″mytable¥″ のように指定します。大/小文字の区別がある値が別の大/小文字の区別がある値によって修飾される場合、2 つの値を個別に区切ります。例えば、″myschema″.″mytable″ のように指定します。ストリングを二重引用符で囲みます。例えば、″select * from newtable″ のようにします。

db2se コマンドを実行すると、コマンドに対応するストアード・プロシージャーが呼び出され、要求した操作が実行されます。

db2se コマンドの概要

以下の表には、Spatial Extender のセットアップと空間データを使用するプロジェクトの作成に関連するタスクを実行するために、どのような db2se コマンドを出すべきかが示されています。また、db2se コマンドの例が示されており、権限とコマンド固有のパラメーターに関する説明もあります。タスクの右側の欄には、ストアード・プロシージャーについての情報に対するリンクまたは参照があります。このストアード・プロシージャーは、コマンドを出すと呼び出されます。ストアード・プロシージャーを使用する権限は、コマンドを使用する権限と同じです。また、コマンドとストアード・プロシージャーは、同じパラメーターを共用します。権限とパラメーターの意味についての詳細は、参照で示されたセクションを調べてください。

表 6. タスクに対応した db2se コマンド

タスク	コマンドと例
座標システムの作成	db2se create_cs コマンド固有のパラメーターと必要な権限は、db2gse.ST_create_coordsys ストアード・プロシージャーのものと同じです。 次の例では、『mycoordsys』という名前の座標システムを作成します。 `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]]`

表 6. タスクに対応した *db2se* コマンド *(続き)*

タスク	コマンドと例
空間参照系の作成	db2se create_srs コマンド固有のパラメーターは、ストアード・プロシージャーのものと同じです。権限は必要ありません。 次の例では、『mysrs』という名前の空間参照系を作成します。 `db2se create_srs mydb -srsName ¥"mysrs¥" -srsID 100 -xScale 10 -coordsysName ¥"GCS_North_American_1983¥"`
空間参照系のドロップ	db2se drop_srs コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次の例では、『mysrs』という名前の空間参照系をドロップします。 `db2se drop_srs mydb -srsName ¥"mysrs¥"`
座標システム定義の削除	db2se drop_cs コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次の例では、『mycoordsys』という名前の座標システムをドロップします。 `db2se drop_cs mydb -coordsysName ¥"mycoordsys¥"`
データを自動的にジオコーディングするセットアップを使用不可にする	db2se disable_autogc コマンド固有のパラメーターと必要な権限は、db2gse.ST_disable_autogeocoding ストアード・プロシージャーのものと同じです。 次の例では、表 MYTABLE の MYCOLUMN という名前のジオコーディングされた列に対する自動ジオコーディングを使用不可にします。 `db2se disable_autogc mydb -tableName ¥"mytable¥" -columnName ¥"mycolumn¥"`
空間操作用にデータベースを使用可能にする	db2se enable_db コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 ページ・サイズが 8 KB で、最小で 500 ページのサイズがある SYSTEM TEMPORARY 表スペースがあることを確認します。そのような表スペースがない場合、作成方法の詳細について「データベース: *管理の概念および構成リファレンス*」の『TEMPORARY 表スペースの作成』を参照してください。 これは、db2se enable_db コマンドを正常に実行するための要件です。 次の例では、空間操作用に、MYDB という名前のデータベースを使用可能にします。 `db2se enable_db mydb`

表 6. タスクに対応した *db2se* コマンド *(続き)*

タスク	コマンドと例
形状ファイルへのデータのエクスポート	db2se export_shape コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次の例では、MYCOLUMN と名付けられた空間列とそれに関連した表 MYTABLE が、 myshapefile という名前の形状ファイルにエクスポートされます。 `db2se export_shape mydb -fileName /home/myaccount/myshapefile -selectStatement "select * from mytable"`
形状ファイルのインポート	db2se import_shape コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次のコマンドは、 myfile という名前の形状ファイルを MYTABLE という名前の表にインポートします。このインポート中に、myfile 内の空間データが、MYCOLUMN という名前の MYTABLE 列に挿入されます。 `db2se import_shape mydb -fileName ¥"myfile¥" -srsName NAD83_SRS_1 -tableName ¥"mytable¥" -spatialColumnName ¥"mycolumn¥"`
ジオコーダーの登録	db2se register_gc コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次の例では、『myschema.myfunction』という名前の関数によってインプリメントされた、 『mygeocoder』という名前のジオコーダーが登録されます。 `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"`
空間列の登録	db2se register_spatial_column コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次の例では、表 MYTABLE の MYCOLUMN という名前の空間列を、空間参照系『USA_SRS_1』に登録します。 `db2se register_spatial_column mydb -tableName ¥"mytable¥" -columnName ¥"mycolumn¥" -srsName USA_SRS_1`

表 6. タスクに対応した *db2se* コマンド *(続き)*

タスク	コマンドと例
データベースを空間操作に使用できるようにするリソースの除去	db2se disable_db コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次の例では、空間操作にデータベース MYDB を使用できるようにするリソースを除去します。 `db2se disable_db mydb`
ジオコーディング操作用セットアップの除去	db2se remove_gc_setup コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次の例では、表 MYTABLE 中の MYCOLUMN という名前の空間列に適用されるジオコーディング操作用のセットアップを除去します。 `db2se remove_geocoding_setup mydb -tableName ¥"mytable¥" -columnName ¥"mycolumn¥"`
バッチ・モードでのジオコーダーの実行	db2se run_gc コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次の例では、表 MYTABLE 中の MYCOLUMN という名前の列にデータを入れるために、ジオコーダーをバッチ・モードで実行します。 `db2se run_gc mydb -tableName ¥"mytable¥" -columnName ¥"mycolumn¥"`
自動的に実行されるジオコーダーのセットアップ	db2se enable_autogeocoding コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次の例では、表 MYTABLE の MYCOLUMN という名前の列に対して、自動ジオコーディングをセットアップします。 `db2se enable_autogeocoding mydb -tableName ¥"mytable¥" -columnName ¥"mycolumn¥"`
ジオコーディング操作のセットアップ	db2se setup_gc コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次の例では、表 MYTABLE の MYCOLUMN という名前の空間列にデータを入れるためにジオコーディング操作をセットアップします。 `db2se setup_gc mydb -tableName ¥"mytable¥" -columnName ¥"mycolumn¥" -geocoderName ¥"db2se_USA_GEOCODER¥" -parameterValues "address,city,state,zip,2,90,70,20,1.1,'meter',4.." -autogeocodingColumns address,city,state,zip commitScope 10`

表 6. タスクに対応した *db2se* コマンド (続き)

タスク	コマンドと例

形状ファイルとその説明に関する情報の表示	db2se shape_info
	
	このコマンドを使用するには、以下のことが必要です。
	• コマンドで参照されるファイルを読み取るための許可を持っている。
	• このファイルを含むデータベースに接続することができる (*–database* パラメーターを使用して、指定したデータベースで互換性のある座標システムおよび空間参照系をシステムが検索するよう指定する場合)。
	
	次の例では、現行ディレクトリーにある myfile という名前の形状ファイルに関する情報を表示します。
	`db2se shape_info -fileName myfile`
	
	次の例では、 offices という名前のサンプル UNIX 形状ファイルに関する情報を表示します。 *–database* パラメーターによって、指定したデータベース (この場合は MYDB) 内の互換性のある座標システムおよび空間参照系が、すべて検出されます。
	`db2se shape_info`
	`-fileName ~/sqllib/samples/extenders/spatial/data/offices`
	`-database myDB`

ジオコーダーの登録抹消	db2se unregister_gc
	
	コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。
	
	次の例では、『mygeocoder』という名前のジオコーダーの登録を解除します。
	`db2se unregister_gc mydb -geocoderName ¥"mygeoco`
	`der¥"`

空間列の登録抹消	db2se unregister_spatial_column
	
	コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。
	
	次の例では、表 MYTABLE の MYCOLUMN という名前の空間列の登録を抹消します。
	`db2se unregister_spatial_column mydb -tableName`
	`¥"mytable¥" -columnName ¥"mycolumn¥"`

座標システム定義の更新	db2se alter_cs
	
	コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。
	
	次の例では、『mycoordsys』という名前の座標システムの定義を新規組織名で更新します。
	`db2se alter_cs mydb -coordsysName ¥"mycoordsys¥"`
	`-organization myNeworganizationb -tableName`
	`¥"mytable¥"`

表 6. タスクに対応した *db2se* コマンド (続き)

タスク	コマンドと例
空間参照系の定義の更新	db2se alter_srs コマンド固有のパラメーターと必要な権限は、ストアード・プロシージャーのものと同じです。 次の例では、『mysrs』という名前の空間参照系を、異なる xOffset と記述により変更します。 `db2se alter_srs mydb -srsName ¥"mysrs¥" -xOffset 35 -description "This is my own spatial reference system."`

db2se upgrade コマンド

db2se upgrade コマンドは、空間操作が可能なデータベースをバージョン 8、バージョン 9.1、またはバージョン 9.5 からバージョン 9.7 にアップグレードします。

このコマンドは、アップグレードを実行するために空間インデックスのドロップや再作成を行う場合があります。そのため、使用する表のサイズに応じて、非常に長い時間がかかることがあります。例えば、使用するデータが 32 ビット・インスタンスから 64 ビット・インスタンスに移動する場合、または DB2 UDB バージョン 8 フィックスパック 6 以前からアップグレードする場合、インデックスがドロップおよび再作成されます。

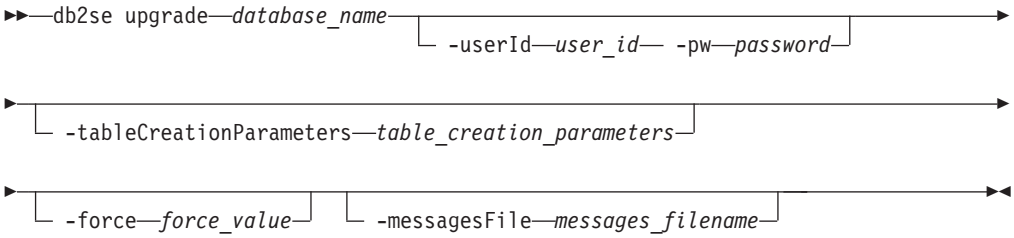
ヒント: オプション `-force 0` を使用して db2se upgrade コマンドを実行し、メッセージ・ファイルを指定して、追加のアップグレード処理なしでアップグレードする必要がある索引を判別します。

許可

アップグレードする空間操作可能なデータベースに関する DBADM 権限または DATAACCESS 権限。

コマンド構文

db2se upgrade コマンド



コマンド・パラメーター

それぞれの意味を説明します。

database_name
アップグレードするデータベースの名前。

-userId user_id
アップグレードするデータベースに対して、DATAACCESS 権限をもつデータベースのユーザー ID。

-pw password
ユーザー・パスワード。

-tableCreationParameters table_creation_parameters
Spatial Extender カタログ表の作成に使用されるパラメーター。

-force force_value

- 0: デフォルト値。データベースのアップグレードを試行しますが、ビュー、関数、トリガー、空間インデックスなどのユーザー定義オブジェクトが Spatial Extender オブジェクトを参照する場合は停止します。
- 1: アプリケーション定義オブジェクトを自動的に保管およびリストア。必要に応じて空間インデックスを保管およびリストアします。
- 2: アプリケーション定義オブジェクトを自動的に保管およびリストア。空間インデックス情報を保管しますが、空間インデックスを自動的にリストアしません。

-messagesFile messages_filename
アップグレード・アクションのレポートが入ったファイル名。指定するファイル名はサーバー上の完全修飾ファイル名でなければなりません。

ヒント: アップグレード時の問題をトラブルシューティングするために、このパラメーターを指定してください。

制約事項: 既存のファイルは指定できません。

使用上の注意

db2se upgrade コマンドは、いくつかの条件を検証して、それらの条件のいずれかが真でなければ、次のうちの 1 つ以上のエラーを戻します。

- データベースは、現在、空間操作可能になっていません。
- データベース名が無効です。
- データベースに対して他の接続が存在します。アップグレードを続行できません。
- 空間カタログが矛盾しています。
- ユーザーは許可されていません。
- パスワードが無効です。
- アップグレードできないユーザー・オブジェクトがあります。

ページ・サイズが 8 KB で、最小で 500 ページのサイズがある SYSTEM TEMPORARY 表スペースがあることを確認します。そのような表スペースがない場合、作成方法の詳細について「データベース: *管理の概念および構成リファレンス*」の『TEMPORARY 表スペースの作成』を参照してください。 これは、db2se upgrade コマンドを正常に実行するための要件です。

db2se migrate コマンド

db2se migrate コマンドによって、空間操作が可能なデータベースをバージョン 9.7 にマイグレーションできます。このコマンドは推奨されません。将来のリリースで除去される予定です。代わりに db2se upgrade コマンドを使用します。

このコマンドは、マイグレーションを実行するために空間インデックスのドロップや再作成を行う場合があります。そのため、使用する表のサイズに応じて、非常に長い時間がかかることがあります。例えば、使用するデータが 32 ビット・インスタンスから 64 ビット・インスタンスに移動する場合、または DB2 バージョン 8 フィックスパック 6 以前からアップグレードする場合、インデックスがドロップおよび再作成されます。

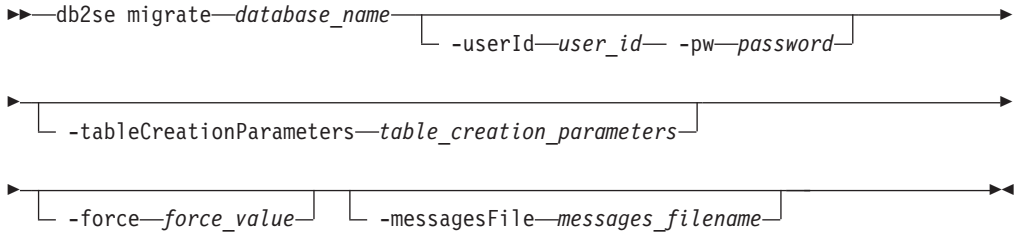
ヒント: オプション `-force 0` を使用して db2se migrate コマンドを実行し、メッセージ・ファイルを指定して、追加のマイグレーション処理なしでマイグレーションする必要がある索引を判別します。

許可

マイグレーションする空間操作可能なデータベースに関する SYSADM 権限または DBADM 権限。

コマンド構文

db2se migrate コマンド



コマンド・パラメーター

それぞれの意味を説明します。

database_name
マイグレーションするデータベースの名前。

-userId user_id
マイグレーションするデータベースに対して、SYSADM 権限か DBADM 権限かのいずれかをもつデータベースのユーザー ID。

-pw password
ユーザー・パスワード。

-tableCreationParameters table_creation_parameters
Spatial Extender カタログ表の作成に使用されるパラメーター。

-force force_value

- 0: デフォルト値。データベース・マイグレーションを試行しますが、ビュー、関数、トリガー、空間インデックスなどのアプリケーション定義オブジェクトが Spatial Extender オブジェクトに基づいていた場合は停止します。
- 1: アプリケーション定義オブジェクトを自動的に保管およびリストア。必要に応じて空間インデックスを保管およびリストアします。
- 2: アプリケーション定義オブジェクトを自動的に保管およびリストア。空間インデックス情報を保管しますが、空間インデックスを自動的にリストアしません。

-messagesFile messages_filename
マイグレーション・アクションのレポートが入ったファイル名。指定するファイル名はサーバー上の完全修飾ファイル名でなければなりません。

ヒント: マイグレーション時の問題をトラブルシューティングするために、このパラメーターを指定してください。

制約事項: 既存のファイルは指定できません。

マイグレーション中に、以下の 1 つ以上のエラーを受け取る可能性があります。
- データベースは、現在、空間操作可能になっていません。
- データベース名が無効です。
- データベースに対して他の接続が存在します。実行できません。
- 空間カタログが矛盾しています。
- ユーザーは許可されていません。
- パスワードが無効です。
- マイグレーションできないユーザー・オブジェクトがあります。

db2se restore_indexes コマンド

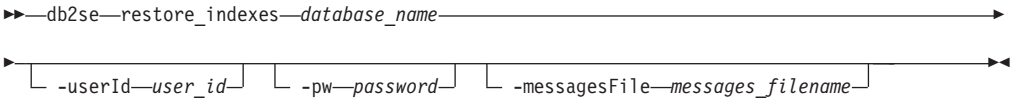
空間操作可能なデータベースへの db2se save_indexes コマンドの実行によって以前に保管した空間インデックスをリストアします。

許可

空間操作可能なデータベースに関する DBADM 権限または DATAACCESS 権限。

コマンド構文

db2se restore_indexes コマンド



コマンド・パラメーター

database_name
マイグレーションするデータベースの名前。

-userId user_id
マイグレーションするデータベースに対して、SYSADM 権限か DBADM 権限かのいずれかをもつデータベースのユーザー ID。

-pw password
ユーザー・パスワード。

-messagesFile messages_filename
マイグレーション・アクションのレポートが入ったファイル名。指定するファイル名はサーバー上の完全修飾ファイル名でなければなりません。

db2se save_indexes コマンド

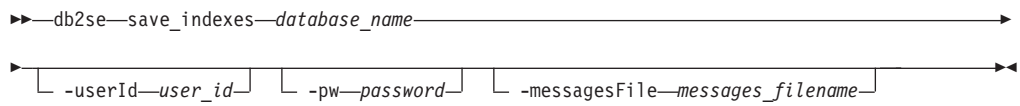
空間操作が可能なデータベースで定義された空間インデックスを保管します。

許可

空間操作可能なデータベースに関する DBADM 権限または DATAACCESS 権限。

コマンド構文

db2se save_indexes コマンド



コマンド・パラメーター

database_name
マイグレーションするデータベースの名前。

-userId user_id
マイグレーションするデータベースに対して、SYSADM 権限か DBADM 権限かのいずれかをもつデータベースのユーザー ID。

-pw password
ユーザー・パスワード。

-messagesFile messages_filename
マイグレーション・アクションのレポートが入ったファイル名。指定するファイル名はサーバー上の完全修飾ファイル名でなければなりません。

第 14 章 アプリケーションの作成およびサンプル・プログラムの使用

この章では、Spatial Extender アプリケーションの作成方法を説明しています。

DB2 Spatial Extender のヘッダー・ファイルを空間アプリケーションに組み込む

DB2 Spatial Extender では、DB2 Spatial Extender のストアード・プロシージャーや関数で使用できる定数を定義したヘッダー・ファイルが提供されています。

推奨:
C または C++ プログラムから DB2 Spatial Extender のストアード・プロシージャーや関数を呼び出す場合は、空間アプリケーションにこのヘッダー・ファイルを組み込んでください。

このタスクは次のように行います。

1. DB2 Spatial Extender アプリケーションが、このヘッダー・ファイルにある必要な定義を使用できるようにします。
   a. DB2 Spatial Extender のヘッダー・ファイルをアプリケーション・プログラムに組み込む。 このヘッダー・ファイルの名前は次のとおりです。

      ```
      db2gse.h
      ```

      ヘッダー・ファイルは *db2path*/include ディレクトリーにあります。*db2path* は、DB2 データベース・システムがインストールされているインストール・ディレクトリーです。
   b. コンパイル・オプションを付けて、`makefile` に include ディレクトリーのパスが指定されていることを確認する。
2. Windows 64 ビット・アプリケーションを Windows 32 ビット・システムで作成する場合は、`samples/extenders/spatial/makefile.nt` ファイル内の DB2_LIBS パラメーターを、64 ビット・アプリケーションに対応できるよう変更します。修正は以下のように行います。

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

アプリケーションから DB2 Spatial Extender のストアード・プロシージャーを呼び出す

DB2 Spatial Extender のストアード・プロシージャーを呼び出すアプリケーション・プログラムを作成する場合は、SQL CALL ステートメントを使用して、ストアード・プロシージャーの名前を指定します。

このタスクについて

空間操作用にデータベースを使用可能にすると、DB2 Spatial Extender のストアード・プロシージャーが作成されます。

© Copyright IBM Corp. 1998, 2009

手順

このタスクは次のように行います。

1. ST_enable_db ストアード・プロシージャーを呼び出して、データベースを空間操作に使用できるようにします。

   次のようにしてストアード・プロシージャー名を指定します。

   ```
   CALL db2gse!ST_enable_db
   ```

   この呼び出しの db2gse! は、DB2 Spatial Extender のライブラリー名を表します。ST_enable_db は、呼び出しに感嘆符を入れる必要のある唯一のストアード・プロシージャーです (db2gse! がこの例です)。.

2. その他の DB2 Spatial Extender ストアード・プロシージャーを呼び出します。次のフォーマットでストアード・プロシージャー名を指定します。db2gse は、すべての DB2 Spatial Extender ストアード・プロシージャーのスキーマ名であり、*spatial_procedure_name* はストアード・プロシージャーの名前です。呼び出しに感嘆符を含めないでください。

   ```
   CALL db2gse.spatial_procedure_name
   ```

   DB2 Spatial Extender のストアード・プロシージャーを以下の表に示します。

表 7. DB2 Spatial Extender ストアード・プロシージャー

ストアード・プロシージャー	説明
ST_alter_coordsys	データベースの座標システムの属性を更新する。
ST_alter_srs	データベース中の空間参照系の属性を更新する。
ST_create_coordsys	データベース中に座標システムを作成する。
ST_create_srs	データベースに空間参照系を作成する。
ST_disable_autogeocoding	DB2 Spatial Extender が、ジオコーディングされた列とこの列に関連するジオコーディングする列との同期処理を停止するように指定する。
ST_disable_db	DB2 Spatial Extender が空間データを格納し、このデータに対して行われる操作をサポートするのに必要なリソースを除去する。
ST_drop_coordsys	データベースから座標システムを削除する。
ST_drop_srs	データベースから空間参照系を削除する。
ST_enable_autogeocoding	DB2 Spatial Extender が、ジオコーディングされた列とこの列に関連するジオコーディングする列との同期化を行うように指定する。
ST_enable_db	データベースが空間データを格納し、操作をサポートするのに必要なリソースを、データベースに提供する。
ST_export_shape	データベース中の選択されたデータを形状ファイルにエクスポートする。
ST_import_shape	形状ファイルをデータベースにインポートする。

表 7. *DB2 Spatial Extender* ストアード・プロシージャー *(続き)*

ストアード・プロシージャー	説明
ST_register_geocoder	DB2 Spatial Extender 製品の一部である DB2SE_USA_GEOCODER 以外のジオコーダーを登録する。
ST_register_spatial_column	空間列を登録し、それに空間参照系を関連付ける。
ST_remove_geocoding_setup	ジオコーディングされる列用のジオコーディング・セットアップをすべて除去する。
ST_run_geocoding	ジオコーダーをバッチ・モードで実行する。
ST_setup_geocoding	ジオコーディングする列をジオコーダーに関連付け、対応するジオコーディング・パラメーター値をセットアップする。
ST_unregister_geocoder	DB2SE_USA_GEOCODER 以外のジオコーダーの登録を抹消する。
ST_unregister_spatial_column	空間列の登録を抹消する。

DB2 Spatial Extender サンプル・プログラム

DB2® Spatial Extender サンプル・プログラムの runGseDemo には、目的が 2 つあります。このサンプル・プログラムは、DB2 Spatial Extender 用のアプリケーション・プログラミングに慣れるため、および、DB2 Spatial Extender のインストールが正しく終了したことを検証するために使用できます。

- UNIX® では、runGseDemo プログラムは以下のパスで見つけることができます。

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

  $HOME は、インスタンス所有者のホーム・ディレクトリーです。
- Windows® では、runGseDemo プログラムは以下のパスで見つけることができます。

  ```
  c:¥Program Files¥IBM¥sqllib¥samples¥extenders¥spatial
  ```

  c:¥Program Files¥IBM¥sqllib は、DB2 Spatial Extender をインストールしたディレクトリーです。

DB2 Spatial Extender の runGseDemo サンプル・プログラムを使用すると、アプリケーション・プログラミングが簡単になります。 このサンプル・プログラムを使用すると、データベースを空間操作で使用可能にできるとともに、そのデータベース内のデータに対する空間分析が行えます。このデータベースには、顧客と洪水地帯の架空情報の入った表が入ります。この情報から、Spatial Extender で実験して、どの顧客が洪水被害に遭う危険があるかを判断することができます。

このサンプル・プログラムでは、以下のことが行えます。

- 空間が使用可能なデータベースの作成と維持に通常必要なステップを知ること。
- アプリケーション・プログラムから空間ストアード・プロシージャーを呼び出す方法を理解すること。

- 独自のアプリケーションにサンプル・コードをカット・アンド・ペーストすること。

DB2 Spatial Extender 用のタスクをコーディングするには、以下のサンプル・プログラムを使用してください。例えば、DB2 Spatial Extender ストアード・プロシージャーを呼び出すためのデータベース・インターフェースを使用するアプリケーションを作成する場合を考えましょう。 このサンプル・プログラムからコードをコピーして、アプリケーションをカスタマイズすることができます。DB2 Spatial Extender のプログラミング・ステップに慣れていない場合は、このサンプル・プログラムを実行すれば、各ステップを詳細に知ることができます。 サンプル・プログラムを実行するための説明については、このトピックの最後にある『関連タスク』を参照してください。

以下の表は、サンプル・プログラムでの各ステップの説明です。各ステップで、アクションを実行し、さらに、多くの場合、そのアクションの反対のアクション、すなわち取り消しを行います。例えば、最初のステップでは、空間データベースを使用可能にし、次に、空間データベースを使用不可にします。このようにして、多くの Spatial Extender ストアード・プロシージャーに慣れていきます。

表 8. DB2 Spatial Extender サンプル・プログラム・ステップ

ステップ	アクションと説明
空間データベースを使用可能または使用不可にする	• 空間データベースを使用可能にする これは、DB2 Spatial Extender を使用するために必要な最初のステップです。 空間操作用に使用可能にされているデータベースには、空間タイプのセット、空間処理関数セット、空間述部セット、新規の索引タイプと 1 組の空間のカタログ表とビューが入っています。 • 空間データベースを使用不可にする このステップは、通常、間違ったデータベースに対して空間機能を使用可能にした場合、あるいは、そのデータベースで空間操作を実行する必要がなくなった場合に、実行します。空間データベースを使用不可にするときは、空間タイプのセット、空間処理関数セット、空間述部セット、新規の索引タイプ、およびそのデータベースに関連付けられている空間カタログ表とビューを除去します。 • 空間データベースを使用可能にする 上記に同じです。

表 8. DB2 Spatial Extender サンプル・プログラム・ステップ (続き)

ステップ	アクションと説明
座標システムを作成またはドロップする	• NORTH_AMERICAN という名前の座標システムを作成する このステップで、データベースに新しい座標システムを作成します。 • NORTH_AMERICAN という名前の座標システムをドロップする このステップで、データベースから座標システム NORTH_AMERICAN をドロップします。 • KY_STATE_PLANE という名前の座標システムを作成する このステップは、新しい座標システム KY_STATE_PLANE を作成します。これは、次のステップで作成される空間参照系で使用されます。
空間参照系を作成またはドロップする	• SRSDEMO1 という名前の空間参照システムを作成する このステップは、座標を解釈するために使用する新しい空間参照系 (SRS) を定義します。この SRS には、空間使用可能データベースの列に保管できるフォーマットで、形状データが収められます。 SRS が特定の空間列に対して登録されると、その空間列に適用される座標が、 CUSTOMERS 表の対応する列に保管できるようになります。 • SRSDEMO1 という名前の SRS をドロップする このステップは、データベースで SRS を必要としなくなった場合に実行します。SRS をドロップするときは、データベースから SRS 定義を取り除きます。
空間表を作成し、データを入れる	• KY_STATE_SRS という名前の SRS を作成する • CUSTOMERS 表を作成する • CUSTOMERS 表にデータを追加する CUSTOMERS 表には、数年にわたってデータベースに保管されているビジネス・データが入っています。 • LOCATION 列を追加して CUSTOMER 表を変更する ALTER TABLE ステートメントによって ST_Point タイプの新しい列 (LOCATION) を追加します。この列には、次のステップでアドレス列をジオコーディングすることによって、データが取り込まれます。 • OFFICES 表を作成する OFFICES 表には、保険会社の各オフィスのセールス・ゾーンが他のデータとともに入っています。この表全体には、後のステップで、非 DB2 データベースから属性データが入れられます。その後続のステップには、形状ファイルから OFFICES 表に属性データをインポートする作業も入っています。

表 8. DB2 Spatial Extender サンプル・プログラム・ステップ (続き)

ステップ	アクションと説明
列にデータを入れる	• KY_STATE_GC という名前のジオコーダーによって、CUSTOMERS 表の LOCATION 列のためのアドレス・データをジオコーディングする このステップでは、ジオコーダー・ユーティリティーを呼び出して、空間のバッチ・ジオコーディングを行います。バッチ・ジオコーディングは、通常、表の大部分をジオコーディングする、または再ジオコーディングする必要があるときに行われます。 • 空間参照システム KY_STATE_SRS を使用して、形状ファイルから以前に作成した OFFICES 表をロードする このステップは、既存の空間データを形状ファイルの形式で OFFICES 表にロードします。OFFICES 表が存在するので、LOAD ユーティリティーは、既存の表に新しいレコードを追加します。 • 空間参照システム KY_STATE_SRS を使用して、形状ファイルから FLOODZONES 表を作成し、ロードする このステップは、形状ファイルの形式で FLOODZONES 表に既存データをロードします。表はまだ存在しないので、LOAD ユーティリティーは、ロードする前に表を作成します。 • 空間参照システム KY_STATE_SRS を使用して、形状ファイルから REGIONS 表を作成し、ロードする
ジオコーダーを登録または登録を抹消する	• SAMPLEGC という名前のジオコーダーを登録する • SAMPLEGC という名前のジオコーダーの登録を抹消する • ジオコーダー KY_STATE_GC を登録する 以上のステップで、SAMPLEGC という名前のジオコーダーを登録および登録を抹消して、次に新しいジオコーダー KY_STATE_GC を作成し、サンプル・プログラムで使用できるようにします。
空間インデックスを作成する	• CUSTOMERS 表の LOCATION 列の空間グリッド索引を作成する • CUSTOMERS 表の LOCATION 列の空間グリッド索引をドロップする • CUSTOMERS 表の LOCATION 列の空間グリッド索引を作成する • OFFICES 表の LOCATION 列の空間グリッド索引を作成する • FLOODZONES 表の LOCATION 列の空間グリッド索引を作成する • REGIONS 表の LOCATION 列の空間グリッド索引を作成する 以上のステップでは、CUSTOMERS、OFFICES、 FLOODZONES および REGIONS 表の空間グリッド索引を作成します。

表 8. *DB2 Spatial Extender* サンプル・プログラム・ステップ *(続き)*

ステップ	アクションと説明
自動ジオコーディングを使用可能にする	• ジオコーダー KY_STATE_GC を使用して、CUSTOMERS 表の LOCATION 列のジオコーディングをセットアップする このステップは、CUSTOMERS 表の LOCATION 列をジオコーダー KY_STATE_GC に関連付け、対応するジオコーディング・パラメーターの値をセットアップします。 • CUSTOMERS 表の LOCATION 列に対して自動ジオコーディングを使用可能にする このステップは、ジオコーダーの自動呼び出しをオンにします。自動ジオコーディングを使用すると、CUSTOMERS 表の LOCATION、STREET、CITY、STATE および ZIP の各列が、後続の挿入および更新操作で互いに同期がとられます。
CUSTOMERS 表に対する挿入、更新、削除操作を実行する	• 異なる番地 (street) を持つレコードをいくつか挿入する • 新しい住所でレコードをいくつか更新する • 表からすべてのレコードを削除する これらのステップは、CUSTOMERS 表の、STREET、CITY、STATE および ZIP 列に対する挿入、更新、削除の操作を示すものです。自動ジオコーディングが使用可能にされると、これらの列に挿入または更新されたデータは、自動的に LOCATION 列にもジオコーディングされます。このプロセスは、前のステップで使用可能にされています。
自動ジオコーディングを使用不可にする	• CUSTOMERS 表の LOCATION 列に対する自動ジオコーディングを使用不可にする • CUSTOMERS 表の LOCATION 列に対するジオコーディングのセットアップを除去する • CUSTOMERS 表の LOCATION 列の空間インデックスをドロップする これらのステップは、次のステップの準備として、ジオコーダーの自動呼び出しと空間インデックスを使用不可にします。次のステップでは、CUSTOMERS 表全体の再ジオコーディングが行われます。 **推奨 :** 大量のジオデータ (地理データ) をロードする場合は、データをロードする前に空間インデックスをドロップし、データのロードが終わった後でもう一度索引を作成してください。

表 8. *DB2 Spatial Extender* サンプル・プログラム・ステップ *(続き)*

ステップ	アクションと説明
CUSTOMERS 表を再ジオコーディングする	• より低いレベルの精度 (100% ではなく 90%) で、CUSTOMERS 表の LOCATION 列を再ジオコーディングする • CUSTOMERS 表の LOCATION 列の空間インデックスを再作成する • 精度レベルを 100% ではなく 90% に下げて、自動ジオコーディングを再び使用可能にする これらのステップは、バッチ・モードでジオコーダーを実行し、空間インデックスを再作成し、新しい精度レベルで自動ジオコーディングを再び使用可能にします。空間管理者がジオコーディング処理での高い障害発生率に気付いたときは、このアクションをとることをお勧めします。精度レベルが 100% に設定されていると、基準データの中に一致アドレスを検出できないためにアドレスのジオコーディングが失敗することがあります。精度レベルを下げると、ジオコーダーは、一致データを検出できやすくなります。バッチ・モードで表を再ジオコーディングした後、自動ジオコーディングがもう一度使用可能にされ、空間インデックスが再作成されます。これによって、これ以降の挿入と更新操作のために、空間インデックスと空間列の増分保守が可能になります。
ビューを作成し、ビューに空間列を登録する	• CUSTOMERS 表と FLOODZONES 表の結合に基づいて、HIGHRISKCUSTOMERS という名前のビューを作成する • ビューの空間列を登録する このステップでは、ビューを作成し、その空間列を登録します。
空間を分析する	• 各地域でサービスする顧客数を調べる (ST_Within) • 同一地域のオフィスと顧客について、各オフィスから一定の範囲内に住んでいる顧客数を調べる (ST_Within、 ST_Distance) • 各地域について、各顧客の平均収入と保険料を調べる (ST_Within) • 各オフィス・ゾーンに重なる洪水地帯の数を調べる (ST_Overlaps) • オフィスがオフィス・ゾーンの中心に位置していると仮定して、特定の顧客のロケーションから最も近いオフィスを見つける (ST_Distance) • 特定の洪水地帯の境界に近いロケーションにある顧客を見つける (ST_Buffer、ST_Intersects) • 特定のオフィスから指定した範囲内のリスクの高い顧客を調べる (ST_Within) 以上のすべてのステップで、sqlRunSpatialQueries ストアード・プロシージャーが使用されます。 これらのステップは、DB2 SQL の関数と空間述部を使用して、空間を分析します。 DB2 照会オプティマイザーは、空間列に関する空間インデックスを活用して、可能なかぎり、照会パフォーマンスを向上させます。

表 8. *DB2 Spatial Extender* サンプル・プログラム・ステップ *(続き)*

ステップ	アクションと説明
空間データを形状ファイルにエクスポートする	• HIGHRISKCUSTOMERS ビューを形状ファイルにエクスポートする このステップでは、HIGHRISKCUSTOMERS ビューを形状ファイルにエクスポートする例を示しています。データベース・フォーマットのデータを別のファイル・フォーマットにエクスポートすることによって、他のツール (ArcExplorer for DB2 など) でその情報が使用できるようになります。 このステップは、runGseDemo.c プログラムに入っていますが、参照用としてコメント化されています。サンプル・プログラムを変更して、エクスポート先の形状ファイルを指定し、サンプル・プログラムを実行し直すことができます。

第 15 章 DB2 Spatial Extender の問題の識別

DB2 Spatial Extender の操作で問題が起こった場合、問題の原因を判別する必要があります。

DB2 Spatial Extender の問題は、以下の方法でトラブルシューティングを行うことができます。

- メッセージ情報を使用して、問題を診断する。
- Spatial Extender のストアード・プロシージャーおよび関数を使用すると、DB2 はストアード・プロシージャーまたは関数が成功したか失敗したかの情報を戻します。戻される情報は、DB2 Spatial Extender の操作に使用したインターフェースにより異なりますが、メッセージ・コード (整数)、メッセージ・テキスト、またはその両方です。
- エラーの診断情報が記録されている DB2 管理通知ファイルは、表示が可能です。
- Spatial Extender の問題が繰り返し起こり、再現可能な場合、問題の診断に役立てるため、DB2 トレース機能を使用することを IBM 担当者がお願いする場合があります。

DB2 Spatial Extender メッセージの解釈方法

以下に示す 4 つの異なるインターフェースを介して、DB2® Spatial Extender で作業を行うことができます。

- DB2 Spatial Extender ストアード・プロシージャー
- DB2 Spatial Extender 関数
- DB2 Spatial Extender コマンド行プロセッサー (CLP)
- DB2 コントロール・センター

要求した空間操作が正常に完了したか、エラーになったかをユーザーが判断するのを助けるために、これらのインターフェースはすべて、DB2 Spatial Extender メッセージを戻します。

後の表は、次のサンプル DB2 Spatial Extender メッセージ・テキストの各部を説明しています。

```
GSE0000I: 操作が正しく完了しました。
```

表 9. DB2 Spatial Extender メッセージ・テキストの各部

メッセージ・テキスト部分	説明
GSE	メッセージ ID。DB2 Spatial Extender メッセージはすべて、3 文字の接頭部 GSE で始まります。
0000	メッセージ番号。0000 から 9999 までの 4 桁の数字。

© Copyright IBM Corp. 1998, 2009

表 9. *DB2 Spatial Extender* メッセージ・テキストの各部 *(続き)*

メッセージ・テキスト部分	説明
I	メッセージ・タイプ。メッセージの重大度を示す単一の文字。 **C** 重大エラー・メッセージ **N** 重大でないエラー・メッセージ **W** 警告メッセージ **I** 通知メッセージ
操作が正しく完了しました。	メッセージの説明。

メッセージ・テキストに表示される説明は、簡単な説明です。詳細な説明と、問題の回避または訂正についての提案を含む、メッセージについての追加情報を検索することができます。この追加情報を表示するには、次のようにします。

1. オペレーティング・システムのコマンド・プロンプトを開きます。
2. メッセージ ID とメッセージ番号を指定して、DB2 ヘルプ・コマンドを入力し、そのメッセージについての追加情報を表示します。 例えば、以下のように指定します。

   ```
   DB2 "? GSEnnnn"
   ```

   ここで *nnnn* はメッセージ番号を表します。

GSE メッセージ ID およびメッセージ・タイプを示す文字は、大文字または小文字で入力できます。DB2 "? GSE0000I" と入力しても、db2 "? gse0000i" と入力しても、結果は同じです。

コマンドを入力するときにはメッセージ番号の後の文字を省略することができます。たとえば、DB2 "? GSE0000" と入力しても、DB2 "? GSE0000I" と入力しても結果は同じです。

メッセージ・コードが GSE4107N であると想定します。コマンド・プロンプトに対して DB2 "? GSE4107N" と入力すると、以下の情報が表示されます。

```
GSE4107N グリッド・サイズ値 "<grid-size>" は、使用されている箇所では無効です。

説明: 指定されたグリッド・サイズ <grid-size> が無効です。

 One of the following invalid specifications was made when the grid index
 was created with the CREATE INDEX statement:

   - A number less than 0 (zero) was specified as the grid size for the
     first, second, or third grid level.

   - 0 (zero) was specified as the grid size for the first grid level.

   - The grid size specified for the second grid level is less than the grid
     size of the first grid level but it is not 0 (zero).

   - The grid size specified for the third grid level is less than the grid
     size of the second grid level but it is not 0 (zero).

   - The grid size specified for the third grid level is greater than 0 (zero)
     but the grid size specified for the second grid level is 0 (zero).

ユーザー応答: グリッド・サイズに有効な値を指定してください。
```

```
msgcode: -4107

sqlstate: 38SC7
```

単一の画面に表示するには情報が長すぎる場合で、オペレーティング・システムが **more** 実行可能プログラムおよびパイプをサポートしている場合、次のコマンドを入力します。

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

more プログラムを使用すると、ユーザーが情報を読めるように、データの各画面を表示後に表示を強制的に一時停止します。

DB2 Spatial Extender ストアード・プロシージャー出力パラメーター

DB2® Spatial Extender ストアード・プロシージャーは、DB2 コントロール・センターから Spatial Extender を有効にして使用する場合、または DB2 Spatial Extender CLP (db2se) を使用するときに、暗黙的に 呼び出されます。ストアード・プロシージャーはアプリケーション・プログラムから、または DB2 コマンド行から明示的に 呼び出すことができます。

このトピックでは、ストアード・プロシージャーがアプリケーション・プログラムから、または DB2 コマンド行から明示的に呼び出されるときの問題を診断する方法を説明します。 暗黙的に呼び出されるストアード・プロシージャーを診断するには、DB2 Spatial Extender CLP によって戻されるメッセージまたは DB2 コントロール・センターによって戻されるメッセージを使用します。これらのメッセージについては、別のトピックで説明します。

DB2 Spatial Extender ストアード・プロシージャーにはメッセージ・コード (msg_code) とメッセージ・テキスト (msg_text) の 2 つの出力パラメーターがあります。パラメーター値は、ストアード・プロシージャーの成功または失敗を示します。

msg_code

msg_code パラメーターは、正、負、またはゼロ (0) の整数です。正数は警告に使用されます。負数はエラー (重大および非重大の両方) に使用されます。そしてゼロ (0) は通知メッセージに使用されます。

msg_code の絶対値は、メッセージ番号として msg_text に組み込まれます。例えば、

- msg_code が 0 の場合、メッセージ番号は 0000 です。
- msg_code が –219 の場合、メッセージ番号は 0219 です。負の msg_code は、メッセージが重大または非重大エラーであることを示します。
- msg_code が +1036 の場合、メッセージ番号は 1036 です。正の msg_code 番号は、メッセージが警告であることを示します。

Spatial Extender ストアード・プロシージャーの msg_code 番号は、次の表に示すように 3 つのカテゴリーに分けられます。

表 10. ストアード・プロシージャー・メッセージ・コード

コード	カテゴリー
0000 – 0999	共通メッセージ
1000 – 1999	管理メッセージ
2000 – 2999	インポートおよびエクスポート・メッセージ

msg_text

msg_text パラメーターは、メッセージ ID、メッセージ番号、メッセージ・タイプ、および説明から構成されています。ストアード・プロシージャー msg_text 値の例を以下に示します。

```
GSE0219N   An EXECUTE IMMEDIATE statement
           failed. SQLERROR = "<sql-error>".
```

msg_text パラメーターに表示される説明は、簡略な説明です。詳細な説明と、問題の回避または訂正についての提案を含む、メッセージについての追加情報を検索することができます。

msg_text パラメーターの各部の詳細な説明および、メッセージに関する追加情報を検索する方法については、トピック「DB2 Spatial Extender メッセージを解釈する方法」を参照してください。

アプリケーションでのストアード・プロシージャーの処理

アプリケーションから DB2 Spatial Extender ストアード・プロシージャーを呼び出すと、出力パラメーターとして msg_code と msg_text を受け取ります。 以下のことを行うことができます。

- 出力パラメーター値をアプリケーション・ユーザーに戻すように、アプリケーションをプログラミングする。
- 戻された msg_code 値のタイプに基づいたアクションを実行する。

DB2 コマンド行からのストアード・プロシージャーの処理

DB2 コマンド行から DB2 Spatial Extender ストアード・プロシージャーを呼び出すと、msg_code および msg_text 出力パラメーターを受け取ります。これらの出力パラメーターは、ストアード・プロシージャーの成功または失敗を示します。

データベースに接続していて ST_disable_db ストアード・プロシージャーを呼び出そうとしていると想定します。以下の例では、DB2 CALL コマンドを使用して、空間操作に対してデータベースを使用不可にしています。そしてその出力値結果を示しています。 CALL コマンドの末尾に、強制パラメーター値 0 が 2 つの疑問符と共に使用されていて、msg_code および msg_text 出力パラメーターを表しています。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

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

  出力パラメーターの値
  --------------------------
  パラメーター名: MSGCODE
  パラメーター値: 0
```

```
パラメーター名: MSGTEXT
パラメーター値: GSE0000I 操作が正しく完了しました。

リターン状況 = 0
```

戻された msg_text が GSE2110N であると想定します。DB2 ヘルプ・コマンドを使用して、メッセージについての詳細な情報を表示します。 例えば、以下のように指定します。

```
"? GSE2110"
```

以下の情報が表示されます。

```
GSE2110N    The spatial reference system for the
            geomentry in row "<row-number>" is invalid.
            The spatial reference system's
            numerical identifier is "<srs-id>".

Explanation: In row row-number, the geometry that is
to be exported uses an invalid spatial reference system.
The geometry cannot be exported.

User Response: Correct the indicated geometry or
exclude the row from the export operation by
modifying the SELECT statement accordingly.

msg_code: -2110

sqlstate: 38S9A
```

DB2 Spatial Extender 関数メッセージ

DB2® Spatial Extender 関数によって戻されるメッセージは、一般的に SQL メッセージに組み込まれています。メッセージの中の戻された SQLCODE は、エラーが関数に発生したかどうか、または警告が関数に関連しているかどうかを示します。例えば、以下のように指定します。

- SQLCODE –443 (メッセージ番号 SQL0443) は、エラーが関数に発生したことを示します。
- SQLCODE +462 (メッセージ番号 SQL0462) は、警告が関数に関連していることを示します。

下表は、次のサンプル・メッセージの重要な部分を説明しています。

```
DB21034E  The command was processed as an SQL statement because it was
not a valid Command Line Processor command.  During SQL processing it
returned: SQL0443N  Routine "DB2GSE.GSEGEOMFROMWKT"
(specific name "GSEGEOMWKT1") has returned an error
SQLSTATE with diagnostic text "GSE3421N  Polygon is not closed.".
SQLSTATE=38SSL
```

表 11. DB2 Spatial Extender 関数メッセージの重要部分

メッセージ部	説明
SQL0443N	SQLCODE は問題のタイプを示します。

表 11. DB2 Spatial Extender 関数メッセージの重要部分 (続き)

メッセージ部	説明
GSE3421N	DB2 Spatial Extender メッセージ番号およびメッセージ・タイプ。 関数のメッセージ番号は GSE3000 から GSE3999 までの範囲です。さらに、DB2 Spatial Extender 関数を処理したときに、共通メッセージが戻ることもあります。 共通メッセージのメッセージ番号は GSE0001 から GSE0999 までの範囲です。
ポリゴンが閉じていません。	DB2 Spatial Extender メッセージの説明。
SQLSTATE=38SSL	エラーをさらに詳細に識別する SQLSTATE コード。SQLSTATE コードは、各ステートメントまたは行に対して戻されます。 • Spatial Extender 関数エラーに対する SQLSTATE コードは 38S*xx* です。各 *x* は文字または数字です。 • Spatial Extender 関数の警告に対する SQLSTATE コードは 01HS*x* です。*x* は文字または数字です。

SQL0443 エラー・メッセージの例

以下に示すように、ポリゴンの値を表 POLYGON_TABLE に挿入しようとしていると想定します。

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

この結果は、エラー・メッセージが出ます。なぜなら、ポリゴンを閉じる終了値を指定していないからです。戻されるエラー・メッセージは次のようになります。

```
DB21034E  The command was processed as an SQL statement because it was
not a valid Command Line Processor command.  During SQL processing it
returned: SQL0443N  Routine "DB2GSE.GSEGEOMFROMWKT"
(specific name "GSEGEOMWKT1") has returned an error
SQLSTATE with diagnostic text "GSE3421N  Polygon is not closed.".
SQLSTATE=38SSL
```

SQL メッセージ番号 SQL0443N は、エラーが発生したことを示し、メッセージには Spatial Extender メッセージ・テキスト「GSE3421N ポリゴンが閉じていません。」が含まれています。

このタイプのメッセージを受け取った場合は、次のようにします。

1. DB2 または SQL エラー・メッセージ内で、GSE メッセージ番号を探し出します。
2. DB2 ヘルプ・コマンド (DB2 ?) を使用して、Spatial Extender メッセージの説明およびユーザー応答を見ます。上記の例を使用した場合、次のコマンドをオペレーティング・システムのコマンド行プロンプトに入力します。

   ```
   DB2 "? GSE3421"
   ```

メッセージが、詳細説明および推奨ユーザー応答とともに繰り返されます。

DB2 Spatial Extender CLP メッセージ

DB2® Spatial Extender CLP (db2se) は以下のものに対してメッセージを戻します。

- ストアード・プロシージャー (暗黙的に呼び出された場合)。
- 形状情報 (DB2 Spatial Extender CLP から **shape_info** サブコマンド・プログラムを呼び出した場合)。 この場合は通知メッセージとなります。
- マイグレーション操作。
- クライアントとの間の形状のインポートおよびエクスポート操作。

DB2 Spatial Extender CLP によって戻されるストアード・プロシージャー・メッセージの例

DB2 Spatial Extender CLP によって戻されるメッセージの大部分は、DB2 Spatial Extender ストアード・プロシージャーに対するものです。 DB2 Spatial Extender CLP からストアード・プロシージャーを呼び出すと、ストアード・プロシージャーの成功または失敗を示すメッセージ・テキストを受け取ります。

メッセージ・テキストは、メッセージ ID、メッセージ番号、メッセージ・タイプ、および説明から構成されています。例えば、コマンド `db2se enable_db testdb` を使用してデータベースを使用可能にする場合、Spatial Extender CLP によって戻されるメッセージ・テキストは次のようになります。

```
Enabling database. Please wait ...

GSE1036W   The operation was successful.  But
           values of certain database manager and
           database configuration parameters
           should be increased.
```

同様に、例えば、コマンド `db2se disable_db testdb` を使用してデータベースを使用不可にした場合、Spatial Extender CLP によって戻されるメッセージ・テキストは次のようになります。

```
GSE0000I 操作が正しく完了しました。
```

メッセージ・テキストに表示される説明は、簡単な説明です。詳細な説明と、問題の回避または訂正についての提案を含む、メッセージについての追加情報を検索することができます。この情報を検索するためのステップ、およびメッセージ・テキストの各部を解釈する方法の詳細については、別のトピックで説明します。

アプリケーション・プログラムから、または DB2 コマンド行からストアード・プロシージャーを呼び出す場合、別のトピックで出力パラメーターの診断について説明しています。

Spatial Extender CLP によって戻される形状情報メッセージの例

`office` という名前の形状ファイルの情報を表示することを決定したと想定します。Spatial Extender CLP (db2se) から、次のコマンドを発行します。

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

以下が表示される情報の例です。

```
Shape file information
----------------------
File code                  = 9994
File length (16-bit words) = 484
Shape file version         = 1000
Shape type                 = 1 (ST_POINT)
Number of records          = 31

Minimum X coordinate = -87.053834
Maximum X coordinate = -83.408752
Minimum Y coordinate = 36.939628
Maximum Y coordinate = 39.016477
Shapes do not have Z coordinates.
Shapes do not have M coordinates.

Shape index file (extension .shx) is present.

Attribute file information
--------------------------
dBase file code                = 3
Date of last update            = 1901-08-15
Number of records              = 31
Number of bytes in header      = 129
Number of bytes in each record = 39
Number of columns              = 3

Column Number  Column Name      Data Type       Length  Decimal

           1   NAME             C ( Character)      16        0
           2   EMPLOYEES        N (   Numeric)      11        0
           3   ID               N (   Numeric)      11        0

Coordinate system definition: "GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]]"
```

Spatial Extender CLP によって戻されるアップグレード・メッセージの例

マイグレーション操作を実行するコマンドを呼び出すと、その操作の成功または失敗を示すメッセージが戻されます。

空間操作が可能なデータベース *mydb* を、以下のコマンドを使用してアップグレードするとします。

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

Spatial Extender CLP によって戻されるメッセージ・テキストは次のようになります。

```
データベースをアップグレードしています。お待ちください...
GSE0000I 操作が正しく完了しました。
```

DB2 コントロール・センター・メッセージ

DB2 Spatial Extender メッセージ

DB2 コントロール・センターを通して DB2® Spatial Extender で作業をした場合、「DB2 メッセージ」ウィンドウにメッセージが表示されます。表示されるメッセージのほとんどは、DB2 Spatial Extender メッセージです。 時には、SQL メッセージを受け取る場合もあります。SQL メッセージは、エラーがライセンス交付やロッキ

ングに関係している場合、または DAS サービスが使用不可である場合に戻されます。以下のセクションでは、DB2 Spatial Extender メッセージおよび SQL メッセージが、DB2 コントロール・センターでどのように表示されるかの例について説明しています。

コントロール・センターを通じて DB2 Spatial Extender メッセージを受け取る場合、メッセージ・テキスト全体が「DB2 メッセージ 」ウィンドウのテキスト域に表示されます。例えば、次のようになります。

```
GSE0219N   An EXECUTE IMMEDIATE statement
           failed. SQLERROR = "<sql-error>".
```

SQL メッセージ

コントロール・センターを通じて DB2 Spatial Extender に関する SQL メッセージを受け取る場合、

- メッセージ ID、メッセージ番号、およびメッセージ・タイプが「DB2 メッセージ」ウィンドウの左側に表示されます。例えば SQL0612N のように表示されます。
- メッセージ・テキストが「DB2 メッセージ」ウィンドウのテキスト域に表示されます。

「DB2 メッセージ」ウィンドウに表示されるメッセージ・テキストには SQL メッセージ・テキストおよび SQLSTATE が含まれる場合があり、メッセージ・テキストと、詳細説明およびユーザー応答が含まれる場合もあります。

SQL メッセージ・テキストおよび SQLSTATE が含まれている SQL メッセージの例は、次のようになります。

```
[IBM][CLI Driver][DB2/NT] SQL0612N "<name>" は重複名です。SQLSTATE=42711
```

メッセージ・テキストと、詳細説明およびユーザー応答が含まれている SQL メッセージの例は、次のようになります。

```
SQL8008N
The product "DB2 Spatial Extender" does not have a valid
license key installed and the evaluation period has expired.

Explanation:

A valid license key could not be found and the evaluation
period has expired.

User Response:

Install a license key for the fully entitled version of the
product.  You can obtain a license key for the product by
contacting your IBM® representative or authorized dealer.
```

db2trc コマンドによる DB2 Spatial Extender の問題のトレース

繰り返し発生し、再現可能な DB2 Spatial Extender 問題を抱えている場合、DB2 トレース機能を使用して、その問題についての情報を収集することができます。

DB2 トレース機能は、**db2trc** システム・コマンドによってアクティブにすることができます。DB2 トレース機能によって以下のことが行えます。

- イベントをトレースする
- トレース・データをファイルへダンプする
- トレース・データを読み取り可能フォーマットにフォーマットする

制約事項:

- DB2 テクニカル・サポート担当者から依頼があった場合にのみ、この機能をアクティブにしてください。
- UNIX オペレーティング・システムでは、DB2 インスタンスをトレースするには SYSADM、SYSCTRL、または SYSMAINT の権限をもっている必要があります。
- Windows オペレーティング・システムでは、特別な権限は必要ありません。

このタスクは次のように行います。

1. 他のすべてのアプリケーションをシャットダウンします。
2. トレースをオンにします。

   DB2 のテクニカル・サポート担当者が、このステップのための特定のパラメーターを提供します。基本コマンドは次のとおりです。

   ```
   db2trc on
   ```

   制約事項: db2trc コマンドは、オペレーティング・システムのコマンド・プロンプトで、またはシェル・スクリプト内に入力する必要があります。 DB2 Spatial Extender コマンド行インターフェース (db2se)、または DB2 CLP では使用できません。

   メモリー、またはファイルにトレースすることができます。トレースの望ましい方式は、メモリーにトレースすることです。再現させる問題がワークステーションを止め、トレースのダンプができなくなる場合は、ファイルにトレースしてください。
3. 問題を再現します。
4. 問題が発生したら直ちにトレースをファイルにダンプします。

   例えば、以下のように指定します。

   ```
   db2trc dump january23trace.dmp
   ```

   このコマンドは、ユーザーが指定する名前で、現行ディレクトリーの中に、ファイル (*january23trace.dmp*) を作成します。そして、トレース情報をそのファイルにダンプします。ファイル・パスを組み込んで別のディレクトリーを指定することができます。例えば、ダンプ・ファイルを /tmp/spatial/errors ディレクトリーに入れるには、構文は次のようになります。

   ```
   db2trc dump /tmp/spatial/errors/january23trace.dmp
   ```
5. トレースをオフにします。

   例えば、以下のように指定します。

   ```
   db2trc off
   ```
6. ASCII ファイルとしてデータをフォーマットします。 2 つの方法でデータをソートすることができます。

- flw オプションを使用して、プロセスまたはスレッドによってデータをソートします。例えば、以下のように指定します。

  ```
  db2trc flw january23trace.dmp january23trace.flw
  ```

- fmt オプションを使用して、すべてのイベントを時系列にリストします。例えば、以下のように指定します。

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

管理通知ファイル

エラーについての診断情報は、管理通知ファイルに記録されます。この情報は DB2® テクニカル・サポートが問題判別のために使用するためのものです。

管理通知ファイルは、DB2 や DB2 Spatial Extender によって記録されるテキスト情報を含んでいるファイルです。 このファイルは、DIAGPATH データベース・マネージャー構成パラメーターで指定されたディレクトリーにあります。Windows® NT、Windows 2000、および Windows XP システムでは、DB2 管理通知ファイルはイベント・ログの中にあり、Windows Event Viewer を使用して調べることができます。

DB2 がどの情報を管理ログの中に記録するかは、DIAGLEVEL および NOTIFYLEVEL の設定によって決まります。

テキスト・エディターを使用して、問題が発生した疑いのあるマシン上のファイルを表示します。記録されている最新のイベントは、ファイルの一番下にあるイベントです。一般的に、各項目には以下のような部分があります。

- タイム・スタンプ。
- エラーの報告をしているロケーション。アプリケーション ID を使用して、サーバーとクライアントのログにあるアプリケーションに関係する項目を見つけることができます。
- エラーを説明する診断メッセージ (通常「DIA」または「ADM」で始まる)。
- 利用可能なサポート・データ (例えば、SQLCA データ構造や、なんらかの追加ダンプ・ファイルまたはトラップ・ファイルのロケーションを指すポインターなど)。

データベースが正常に動作している場合は、このタイプの情報は重要ではなく、無視できます。

管理通知ファイルは、次第に大きくなります。大きくなりすぎた場合は、バックアップをとって、ファイルを消去します。次にシステムで必要となると、新規のファイルが自動的に生成されます。

第 16 章 DB2 Geodetic Data Management Feature

この章では、DB2 Geodetic Data Management Feature の目的、用途を説明し、測地という概念を説明して、当製品の概要を紹介しています。

DB2 Geodetic Data Management Feature

DB2® Geodetic Data Management Feature では、地球を球体として扱うことができます。 Geodetic Data Management Feature では、他の Spatial Extender 操作と同じデータ・タイプおよび関数を使用して、極地の周囲のデータおよび 180 度の子午線をまたがるデータに対してもシームレスに照会を実行できます。これにより、地球表面の各地点の位置をより正確に表すデータを保守することができます。

Geodetic Data Management Feature という名前は、*測地学 (geodesy)* という学問に由来します。測地学は、地球 (太陽、天球など、楕円によってモデル化されるあらゆる物) の大きさや形状についての学問です。 Geodetic Data Management Feature は、地球表面上に存在するオブジェクトを高精度で取り扱えるよう設計されています。

高精度を確保するために、Geodetic Data Management Feature では、平面の *xy* 座標システムではなく、楕円体地球モデルまたは *測地基準* に基づく緯度および経度の座標システムを使用しています。楕円体モデルを使用すると、平面投影を使用した場合のようなゆがみや誤り、不正確さはなくなります。

空間操作機能ではなく測地操作機能を使用するためには、操作対象のデータに対応する測地参照系を定義する必要があります。この座標システムには、2000000000 から 2000001000 までの範囲の空間参照系 ID (SRID) を持たせます。 DB2 Geodetic Data Management Feature は 318 の定義済み測地空間参照系を提供しています。

DB2 Geodetic Data Management Feature を使用するには、まず DB2 Spatial Extender をインストールする必要があります。Geodetic Data Management Feature を使用可能にするには、ライセンスを別途購入する必要があります。

DB2 Geodetic Data Management Feature を使用する場合と DB2 Spatial Extender を使用する場合

DB2® Spatial Extender と DB2 Geodetic Data Management Feature はいずれも、DB2 データベースの地理情報システム (GIS) データを管理するものです。両者はそれぞれに中核となる技術も解決できる問題も異なっており、互いに補完し合う関係にあります。

- Geodetic Data Management Feature は地球を球体として扱います。そして、楕円体地球モデルに基づいた緯度経度座標システムを使用します。幾何学的計算は、対象となる位置に関係なく正確に行えます。その基礎となっているのは、Geodyssey Limited からライセンス交付を受けた Hipparchus ライブラリーです。 測地情報の詳細については、http://www.geodyssey.com を参照してください。

© Copyright IBM Corp. 1998, 2009

Geodetic Data Management Feature は、地球上の広い (一度の地図投影ではアプリケーションに必要な正確さを十分に得ることができないほどの) 領域をカバーするようなグローバルなデータ・セットやアプリケーションに最適です。

- Spatial Extender は地球を平面の地図として扱います。地球の表面は実際には曲面になっているため、投影によって、それに近い平面の地図を作り出しているということです。この投影によってゆがみが生じます。データのカバーする範囲によっても異なりますが、一般にゆがみは投影領域の端に近づくほど大きくなります。平面の地図への投影は、どのような方法を採っても必ず何らかのゆがみを生じます。Spatial Extender の基礎となっているのは、ESRI からライセンスを取得した、ESRI 形状ライブラリーです。空間情報の詳細については、http://www.esri.com を参照してください。

  Spatial Extender は、特定の地域のみに対応する (投影座標システムで十分正確に表現できる) ローカルなデータ・セットや、位置の正確性があまり重要でないアプリケーションに適しています。例えば、医療保険会社が、特定の都道府県の中の病院や診療所の位置を確かめる、というような場合には向いているでしょう。

測地基準

測地基準は、地球表面についてのデータに使用する参照系の 1 つです。科学が発達し、地球を計測するための手段が増えるのに伴い、こうした参照系はここ何世紀かの間に多数考案されています。測地系の作成にも、平面地図の投影にも、地上での測量、人工衛星による測量の両方が使用されています。

測地基準は、回転楕円 (*回転楕円体* とも呼ばれる) による地球全体の形状の近似を基礎としたものです。回転楕円体とは、楕円を軸の 1 つの周囲で回転させることによってできる 3 ディメンション形状です。

空間のオブジェクトを定義する際には、対応する測地系を指定する必要があります。測地系の指定には、空間参照系 ID (SRID) を使用します。 DB2® Geodetic Data Management Feature のサポートしている測地系であれば、いずれも選択できます。 Geodetic Extender のサポートしている測地系は 2000000000 から 2000001000 までの範囲の SRID を持っています。

- 「DB2 Geodetic Data Management Feature によってサポートされる測地系」には、Geodetic Data Management Feature が提供している 318 の定義済み測地参照系のリストが記載されています。
- 2000000318 から 2000001000 までの範囲の ID を使用すれば、空間参照系を作成して新たな測地系を定義することもできます。

制約事項: 複数の地理空間オブジェクトを引数とする関数は、複数の測地系が同時に使用されていると対応できません。 Geodetic Data Management Feature は、測地系変換を行いません。

測地緯度と測地経度

DB2 Geodetic Data Management Feature 座標参照系では、 *測地緯度* と *測地経度* を使用して、地球上の各地点の相対的な位置を表します。 測地緯度と測地経度は、必ず特定の測地系を基準にして決められます。

測地緯度

あるポイントの測地緯度は、赤道面と、地球表面上のそのポイントを通る法線と交差する垂直線との間の角度で表されます。

測地経度

あるポイントの*測地経度*は、赤道面上の、地球の中心と本初子午線を結ぶ直線 *a* と、そのポイントが位置する子午線と地球の中心を結ぶ直線 *b* の間の角度で表されます。*子午線* は、測地系の表面上を通り、両極間の最短距離を直接結ぶ線です。

図 17 の楕円には、測地緯度と測地経度がどの角度であるかが示されています。地球の形状は楕円なので、測地緯度を計測する直線は、地球の「真の」中心から始まってはいません。

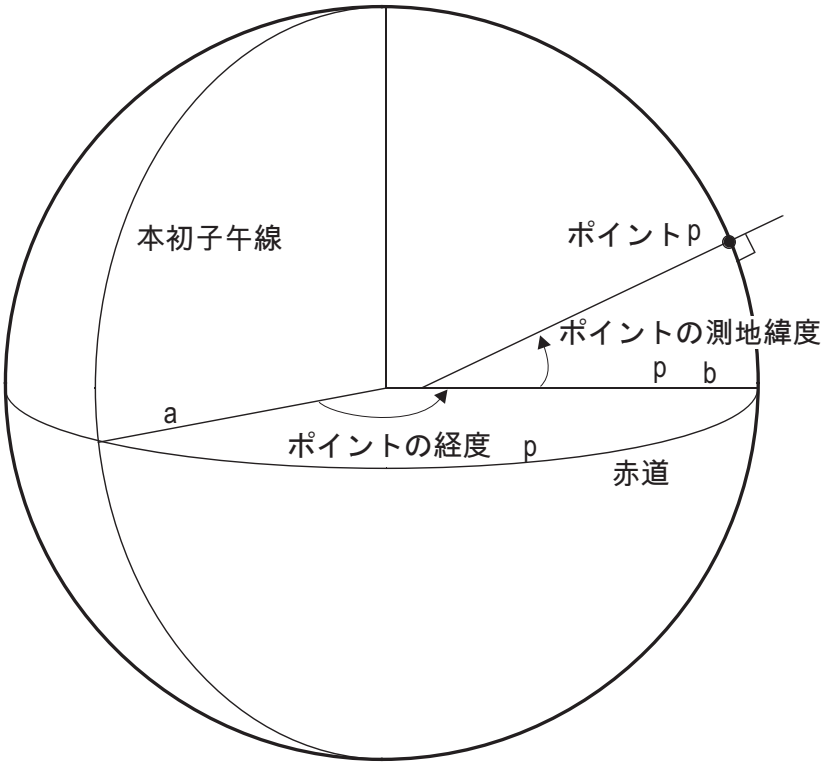


図 17. 測地緯度と測地経度に対応する角度

緯度座標と経度座標は小数部を持つ「度」という単位で表されます。経度は本初子午線 (経度 0°) を基点として、東へ進むと正の値 (180°まで) になり、西へ進むと負の値 (–180°まで) になるので、合計で 360 度ということになります。緯度は、赤道面 (緯度 0°) を基点とし、北極点の方向に進むと正の値 (北極点で 90°になる) になり、南極点の方向に進むと負の値 (南極点で –90°になる) になります。

測地距離

DB2® Geodetic Data Management Feature では、2 ポイント間の距離を*測地線 (geodesic)* に沿って測定します。測地線とは、楕円である地球上で、2 ポイント間の最短距離を結んだ曲線のことです。たとえ 2 つのポイントが同じ緯度上にあったとしても、測地線の緯度は常に一定とは限りません。

直線セグメントは測地線であるとして演算が行われるので、各ポイント間の距離が長い 4 ポイント・ポリゴンが囲む領域は、図 18 に示すように、ユーザーの意図と異なったものになる場合があります。図のポリゴンは、左右の距離が経度にして約 120 度あり、上の 2 ポイントと下の 2 ポイントがそれぞれ同じ緯度にある、という領域をカバーしています。 2 つの経線の間にある測地線は、楕円である地球表面の形状に沿っているため曲線になっています。測地線の中央部分と両端部分では、緯度にして 20 度以上の違いが生じています。

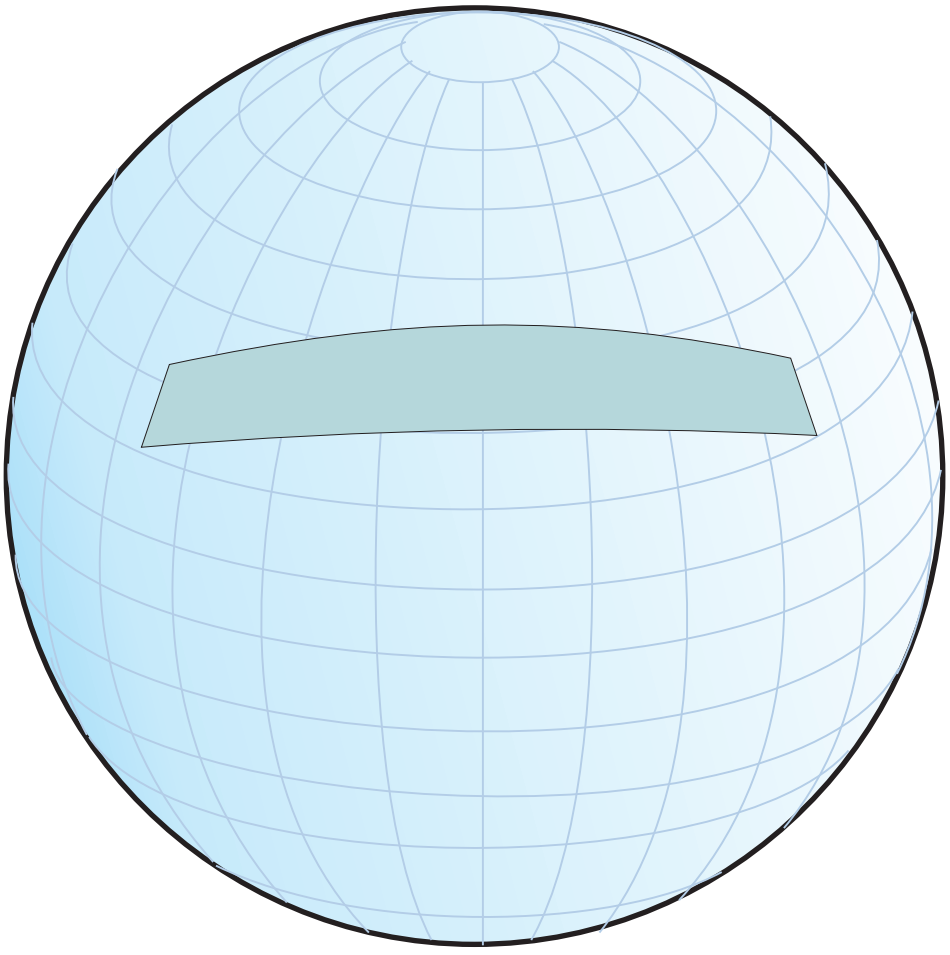


図 18. ポイント間の距離が長いポリゴンの囲む領域

測地線とは異なる線、例えば、緯度が常に一定になる直線セグメントなどを表現するには、中間のポイントを別途追加する必要があります。

測地領域

測地領域 (ポリゴン) とは、ある特定のアプリケーションに対応する特性を持った地球表面上の領域のことです。例としては、ある商品の市場動向に大きな影響を与えると考えられる地域、ある時間に人工衛星から見える地域などがあげられます。

Geodetic Data Management Feature は、クローズされたリングを形成する、一定の順序で並んだ一連のポイントによって領域を定義します。ユーザーがポリゴン中のポイントを指定する順序も意味を持つので注意が必要です。ユーザーが、定義された順序でポリゴンの頂点を 1 つ 1 つたどった場合には、左側の領域がポリゴンの内部ということになります。

図 19 に示すように、1 つ以上のリングに囲まれた領域を定義するには、ST_Polygon データ・タイプが使用できます。ポリゴンを、ポイント (頂点) の緯度座標、経度座標によって定義すると、リングができます。

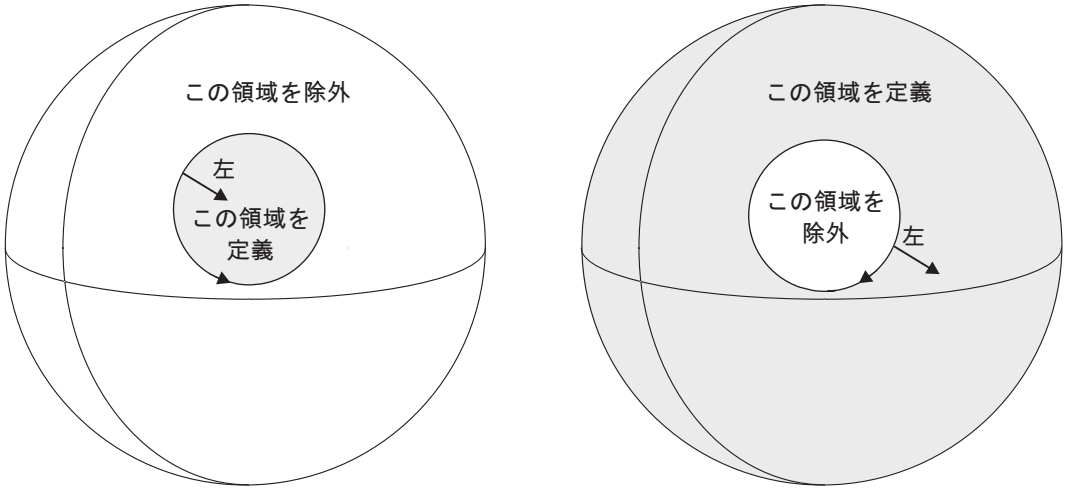


図 19. 領域の定義と除外

リングは、地球表面を、ポリゴンの中と外という 2 つの領域に分割します。 図 19 の左に示したのは、頂点が左回りの順序で指定されたリングで、この場合、指定された頂点の左側に位置するポイントがすべてリングの中にある、ということになります。右に示したのは、頂点が右回りの順序で指定されたリングで、この場合、指定された頂点の左側に位置するポイントがすべてリングの外にある、ということになります。

領域をポリゴンとして定義する場合は、リングの横断時にポリゴンの内部が左側に位置するように、各リングの頂点の順序を指定する必要があります。領域から除外する部分を定義するには、 136 ページの図 20 のように、リングの頂点の順序を右回りに指定します。ポリゴンの内部は、常に頂点の左側にあります。 136 ページの図 20 には、一方がもう一方の中に入っているという 2 つのリングを示してあります。大きい方のリングは、ポリゴンの外側の境界を定義していて、左回りに描かれています。小さい方のリングは、内側の境界を定義していて、右回りに描かれています。

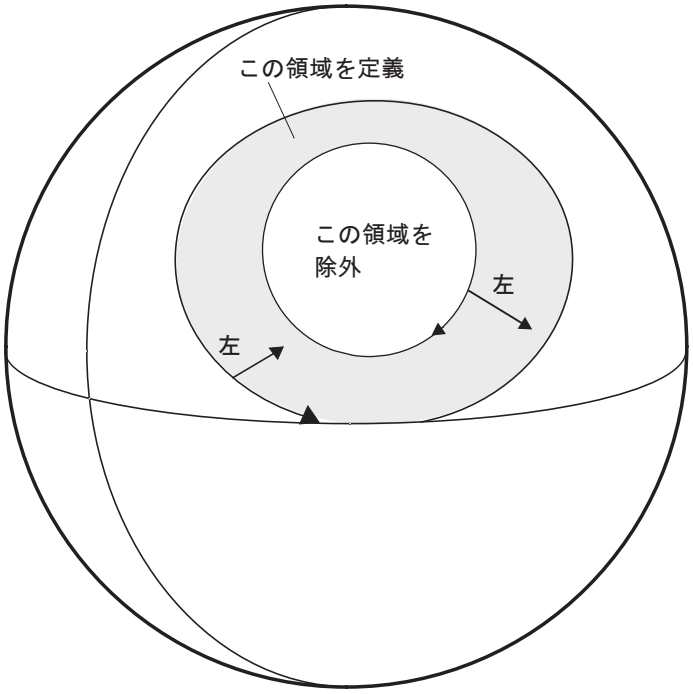


図 20. 複数のリングでの領域の定義

半球より大きいポリゴンを作ると、以下のような警告メッセージが戻されます。本当に意図して大きいポリゴンを作ろうとしている場合には問題ありませんが、小さいポリゴンを作るつもりが、頂点の指定の順序を誤ったために結果的に大きいポリゴンを作ってしまった、という場合もあるために、この警告が送信されます。

```
GSE3733W "Polygon covers more than half the earth. Verify counter-clockwise
  orientation of the vertex points."
```

第 17 章 DB2 Geodetic Data Management Feature のセットアップ

このセクションでは、DB2 Geodetic Data Management Feature のセットアップ、Informix® Geodetic DataBlade® からのマイグレーション、空間列への測地データの追加などについて説明します。

DB2 Geodetic Data Management Feature のセットアップと使用可能化

DB2 Geodetic Data Management Feature を使用可能にする前に、以下のことを行う必要があります。

- DB2 バージョン 9.5 のインストールと構成。

  DB2 Spatial Extender と DB2 Geodetic Data Management Feature をインストールする *前* に、DB2 データベースをインストールする必要があります。DB2 コントロール・センターを使用する場合は、DB2 Administration Server (DAS) を作成し、構成します。DAS を作成し、構成する方法については、「*IBM DB2 管理ガイド: インプリメンテーション*」を参照してください。
- DB2 Spatial Extender のインストールと構成。

  DB2 Geodetic Data Management Feature は、DB2 Spatial Extender と同じライブラリー・コードに統合されます。そのため、Spatial Extender のインストール CD には Geodetic Data Management Feature が含まれています。Spatial Extender のディスク・スペース所要量には、Geodetic Data Management Feature に必要な分も含まれています。ただし、Geodetic Data Management Feature のライセンスを購入して使用可能にするまでは、Geodetic Data Management Feature を使用することはできません。
- DB2 Geodetic Data Management Feature のライセンスの購入。

  DB2 Geodetic Data Management Feature ライセンスを購入すると、Geodetic ライセンス・キーを使用可能にできます。 DB2 Geodetic Data Management Feature を購入する場合は、営業担当者にご連絡ください。

制約事項:

- DB2 Geodetic Data Management Feature は、DB2 バージョン 9.5 Enterprise Server Edition に対してのみライセンスされています。
- Spatial Extender では、本来は曲面である地球表面を平面の地図として扱いますが、DB2 Geodetic Data Management Feature では地球を球体として扱います。Geodetic Data Management Feature をインストールすると、平面地図を使った場合よりも正確に空間データを分析できるようになります。
- DB2 Geodetic Data Management Feature システムは、DB2 データベース・システム、DB2 Spatial Extender、DB2 Geodetic Data Management Feature から構成され、ほとんどのアプリケーションでは、ジオブラウザーも加わります。

© Copyright IBM Corp. 1998, 2009

DB2 Geodetic Data Management Feature を使用可能にするための追加情報や変更情報については、「*DB2 リリース情報*」を参照してください。

DB2 Geodetic Data Management Feature ライセンスを使用可能にした後で、空間列に測地データを入れます。

このタスクは次のように行います。

DB2 Geodetic Data Management Feature ライセンスの使用可能化は、以下のいずれかの方法で行います。

- DB2 コントロール・センターの「ライセンス・センター」を使用する。測地ライセンスの使用可能化の詳細については、DB2 ライセンス・センターのオンライン・ヘルプを参照してください。
- **db2licm** コマンドを実行します。

Informix Geodetic DataBlade から DB2 Geodetic Data Management Feature へのマイグレーション

前提条件

Geodetic DataBlade アプリケーションを、DB2 Geodetic Data Management Feature のデータ・タイプや関数を利用できるよう移植する必要があります。

制約事項

現在 Informix Geodetic DataBlade を使用している場合、以下の条件が満たされていれば、DB2 Geodetic Data Management Feature へのマイグレーションが可能な場合があります。

- データ・タイプとして、 GeoPoint、 GeoLineseg、 GeoString、 GeoRing、 GeoPolygon のみを使用する。
- Geodetic DataBlade 関数の中でも、DB2 Geodetic Data Management Feature に同等、あるいは類似の関数があるもののみを使用する (関数とデータ・タイプの対応関係については下にあげる表を参照)。
- GeoObjects の空間コンポーネントのみを索引に含める。つまり、時間の範囲や高度の範囲は索引に含めない。

IBM Informix Geodetic DataBlade を使用してデータベース中の地理空間オブジェクトを保存、操作している場合は、データやアプリケーションを IBM DB2 Geodetic Data Management Feature に、いくつかの制限はありますがマイグレーションできます。

このタスクは次のように行います。

1. IBM Informix Geodetic DataBlade から IBM DB2 Geodetic Data Management Feature へのマイグレーションは以下の手順で行います。

表 12. Informix Geodetic DataBlade と Geodetic Data Management Feature のデータ・タイプの対応関係

Informix Geodetic DataBlade のデータ・タイプ	DB2 Geodetic Data Management Feature の対応データ・タイプ	類似データ・タイプに関するコメント
GeoEllipse		まず GeoPolygon に変換した後、ST_Polygon にマイグレーションする。
GeoLineseg	ST_LineString	
GeoObject	ST_Geometry	ST_Geometry とそのサブタイプは、GeoAltRange、GeoTimeRange データ・タイプをサポートしていない。
GeoPoint	ST_Point	
GeoPolygon	ST_MultiPolygon、ST_Polygon	ST_MultiPolygon は、各リングに、明示的な閉包ポイントを必要とする。 GeoPolygon に外側のリングがあれば、ST_Polygon にマッピングできる。
GeoRing	ST_LineString	
GeoString	ST_LineString	

以下の Geodetic DataBlade データ・タイプには、対応する Geodetic Data Management Feature のデータ・タイプはありません。

- GeoAltitude
- GeoAltRange
- GeoAngle
- GeoAzimuth
- GeoCoords
- GeoDistance
- GeoEllipse
- GeoLatitude
- GeoLongitude
- GeoTimeRange
- GeoVoronoi

a. SQL ステートメントを、DB2 Geodetic Data Management Feature のデータ・タイプや関数を利用するよう書き直す。

b. データを DB2 Geodetic Data Management Feature にロードあるいはインポートする。

c. Informix ODBC、ESQL/C、JDBC を利用するアプリケーションを書き直す。

145 ページの表 22に、Geodetic DataBlade と Geodetic Data Management Feature の対応クライアントの接続性についてまとめてあります。

表 13. Informix Geodetic DataBlade と Geodetic Data Management Feature の述部関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Data Management Feature の対応関数
Contains	ST_Contains
Inside	ST_Within

表 13. Informix Geodetic DataBlade と Geodetic Data Management Feature の述部関数の対応関係 (続き)

Informix Geodetic DataBlade の関数	DB2 Geodetic Data Management Feature の対応関数
Intersect	ST_Intersects
Outside	ST_Disjoint
Within	ST_Distance

2. 以下の Geodetic DataBlade の述部関数には、対応する Geodetic Data Management Feature の関数はありません。
   - Beyond
   - Equal
   - Nearest

表 14. Informix Geodetic DataBlade と Geodetic Data Management Feature のプロダクション関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Data Management Feature の対応関数	類似の関数に関するコメント
Difference	ST_Difference	ST_Difference はポリゴンに加え、ポイントをサポートしている。
Generalize	ST_Generalize	
Intersection	ST_Intersection	ST_Intersection(line,line) の実行結果は複数ポイントになることがある。 ST_Intersection (line,poly) の実行結果は複数折れ線になることがある。オブジェクトに共通部分がない場合は、戻り値が空になる。
SymDifference	ST_SymDifference	ST_SymDifference はポリゴンに加え、ポイントをサポートしている。
Union	ST_Union	ST_Union はポリゴンに加え、ポイントと線をサポートしている。

表 15. Informix Geodetic DataBlade と Geodetic Data Management Feature のアクセサー関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Data Management Feature の対応関数	類似の関数に関するコメント
Center	ST_MidPoint、ST_PointOnSurface	ST_MidPoint は、線に関しては、ほぼ代用になる。ST_PointOnSurface は、ポリゴンに関しては、ほぼ代用になる。

表 15. Informix Geodetic DataBlade と Geodetic Data Management Feature のアクセサー関数の対応関係 (続き)

Informix Geodetic DataBlade の関数	DB2 Geodetic Data Management Feature の対応関数	類似の関数に関するコメント
Coords	ST_PointN	
ディメンション	ST_Dimension	
HasZValue	ST_Is3d	
IsGeoBox	IS OF 式、あるいは ST_GeometryType を使用する。	
IsGeoCircle	IS OF 式、あるいは ST_GeometryType を使用する。	
IsGeoEllipse	IS OF 式、あるいは ST_GeometryType を使用する。	
IsGeoLineseg	IS OF 式、あるいは ST_GeometryType を使用する。	
IsGeoPoint	IS OF 式、あるいは ST_GeometryType を使用する。	
IsGeoPolygon	IS OF 式、あるいは ST_GeometryType を使用する。	
IsGeoRing	IS OF 式、あるいは ST_GeometryType を使用する。	
IsGeoString	IS OF 式、あるいは ST_GeometryType を使用する。	
Latitude	ST_Y	
Longitude	ST_X	
NPoints	ST_NumPoints	
NRings	ST_NumGeometries、ST_NumInteriorRing	外側のリングの合計数を得るには ST_NumGeometries を使用し、複数ポリゴンのセット中の個々のポリゴンに関しては ST_NumInteriorRings を合計する。
Ring	ST_GeometryN, ST_ExteriorRing, ST_InteriorRingN	ST_GeometryN を ST_ExteriorRing および ST_InteriorRingN と組み合わせて使用する。
SRID	ST_SRID	
Zvalue	ST_Z	

3. 以下の Geodetic DataBlade のアクセサー関数には、対応する Geodetic Data Management Feature の関数はありません。
   - IsLarge
   - IsSmallArea

表 16. Informix Geodetic DataBlade と Geodetic Data Management Feature の修飾関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Data Management Feature の対応関数
SetSRID	ST_SRID

4. 以下の Geodetic DataBlade の修飾関数には、対応する Geodetic Data Management Feature の関数はありません。
   - SetAltRange
   - SetAltRangeZ
   - SetDist
   - SetTimeRange

表 17. Informix Geodetic DataBlade と Geodetic Data Management Feature の測定関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Data Management Feature の対応関数
Area	ST_Area
Distance	ST_Distance
長さ	ST_Length、ST_Perimeter

5. VoronoiResolution 測定関数には、対応する Geodetic Data Management Feature の関数はありません。

表 18. Informix Geodetic DataBlade と Geodetic Data Management Feature のダウン・キャスト関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Data Management Feature の対応関数
GeoBox	SQL TREAT 式を使用する。
GeoCircle	SQL TREAT 式を使用する。
GeoEllipse	SQL TREAT 式を使用する。
GeoLineseg	SQL TREAT 式を使用する。
GeoPoint	SQL TREAT 式を使用する。
GeoPolygon	SQL TREAT 式を使用する。
GeoRing	SQL TREAT 式を使用する。
GeoString	SQL TREAT 式を使用する。

表 19. Informix Geodetic DataBlade と Geodetic Data Management Feature のコンストラクター関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Data Management Feature の対応関数
GeoCoords	ST_Point
GeoPoint	ST_Point

6. 以下の Geodetic DataBlade のコンストラクター関数には、対応する Geodetic Data Management Feature の関数はありません。
   - GeoBox
   - GeoCircle
   - GeoEllipse
   - GeoLineseg

表 20. Informix Geodetic DataBlade と Geodetic Data Management Feature の診断関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Data Management Feature の対応関数
GeoTraceLevel	DB2 Trace Facility
IsValidGeometry	ST_IsValid

7. 以下の Geodetic DataBlade の診断関数には、対応する Geodetic Data Management Feature の関数はありません。
   - GeoInRowSize
   - GeoOutOfRowSize
   - GeoRelease
   - GeoTotalSize
   - GeoTraceLevelSet
   - GeoWarningLevel
   - GeoWarningLevelSet
   - IsValidSDTS

表 21. Informix Geodetic DataBlade と Geodetic Data Management Feature のシステム・カタログ表の対応関係

Informix Geodetic DataBlade のシステム・カタログ表	DB2 Geodetic Data Management Feature の対応カタログ・ビュー
GeoLenUnit	DB2GSE.ST_UNITS_OF_MEASURE
GeoSpatialRef	DB2GSE.SPATIAL_REF_SYS

8. 以下の Geodetic DataBlade システム・カタログ表には、対応する Geodetic Data Management Feature の表やビューはありません。
   - GeoEllipsoid
   - GeoParam
   - GeoVoronoi

9. 以下の Geodetic DataBlade のユーザー設定可能なパラメーター関数に対応する関数は、Geodetic Data Management Feature にはありません。
   - GeoParamSessionGet
   - GeoParamSessionSet
10. 以下の Geodetic DataBlade の AltRange 関数には、対応する Geodetic Data Management Feature の関数はありません。
   - AltRange
   - Bottom
   - Contains
   - Equal
   - Inside
   - Intersect
   - IsAny
   - Outside
   - Top
11. 以下の Geodetic DataBlade の TimeRange 関数には、対応する Geodetic Data Management Feature の関数はありません。
   - Begin
   - Contains
   - End
   - Equal
   - IsAny
   - Inside
   - Intersect
   - Outside
   - TimeRange
12. 以下の Geodetic DataBlade の楕円関数には、対応する Geodetic Data Management Feature の関数はありません。
   - Azimuth
   - Coords
   - Major
   - Minor
13. 以下の Geodetic DataBlade の円関数には、対応する Geodetic Data Management Feature の関数はありません。
   - Coords
   - Radius
14. 以下の Geodetic DataBlade の角度演算関数には、対応する Geodetic Data Management Feature の関数はありません。
   - Divide
   - Minus
   - Negate

- Plus
- Times

15. 以下の Geodetic DataBlade のクライアント接続には、対応する Geodetic Data Management Feature のクライアント接続はありません。
    - Java API
    - LIBMI

表 22. Geodetic DataBlade と DB2 Geodetic Data Management Feature の、対応クライアント接続製品

Informix Geodetic DataBlade のクライアント接続	**DB2 Geodetic Data Management Feature の対応クライアント接続**
ESQLC	SQC
ODBC	ODBC
JDBC	JDBC

空間列に測地データを入れる

空間列を作成し、空間インデックスを作成する列を登録すれば、列に測地データを入れる準備ができたことになります。シェイプ・フォーマットでデータをインポートするか、または以下のデータ・フォーマットで値を挿入または更新して、測地データを入れることができます。

- 形状
- 事前割り当てテキスト (WKT)
- 事前割り当てバイナリー (WKB)
- GML (Geography Markup Language)

制約事項:

- Spatial Extender の場合、データを測地データに変換するためにジオコーダー・コマンドやストアード・プロシージャーを使用することはできません。
- 測地動作に対応するには、2,000,000,000 から 2,000,001,000 までの範囲の SRID を持つ空間参照系を使用します。
- シェイプ・データは、地理座標システムで表現されている必要があります。

測地データをインポートする手順は、空間データと同じです。

第 18 章 測地索引

測地ボロノイ・インデックスを作成すれば、測地データ照会の際のパフォーマンスを向上できます。この章では、以下の項目を説明しています。

- 測地ボロノイ・インデックスについての説明
- ボロノイ・セル構造について、および代替構造をいつ選択するのかについての説明
- 測地ボロノイ・インデックスの作成方法についての説明

測地ボロノイ・インデックス

DB2® Geodetic Data Management Feature は、測地データへのアクセスを高速化する測地ボロノイ・インデックスを提供しています。この索引は、地球表面のボロノイ分割によって、測地データへのアクセスを整理するものです。

Geodetic Data Management Feature は、形状ごとに最小外接円 (MBC) を算出します。MBC とは、測地形状を囲む円のことです。ボロノイ索引では、セル構造中のデータの整理にこの MBC の情報を利用します。ボロノイ索引を利用して検索を行うと、データが整理され、検索対象の領域をおおまかに絞り込むことができるため、目的のオブジェクトを早く見つけ出すことができ、オブジェクト自身について調べる処理もより正確に行えるようになります。ボロノイ索引を利用すれば、該当領域の外にあるオブジェクトについて調べる必要はなくなるため、パフォーマンスの向上につながります。ボロノイ索引を使用しなかった場合、照会は、指定の条件に合うすべてのオブジェクトについて評価しなければならなくなります。

オプティマイザーは、Where 節中に以下の関数を含む照会のすべてについて測地ボロノイ・インデックスの使用を検討します。

- EnvelopesIntersect
- ST_Contains
- ST_Distance
- ST_EnvIntersects
- ST_Intersects
- ST_MBRIntersects
- ST_Within

測地ボロノイ・インデックスを作成する際には、代替ボロノイ・セル構造を選択することもできます。

© Copyright IBM Corp. 1998, 2009

ボロノイ・セル構造

計算を効率よく実行するため、DB2® Geodetic Data Management Feature は、地球表面を小さく処理が容易な、蜂の巣のようなセルに分割します。この再分割は、*ボロノイ分割* と呼ばれており、分割によってできるデータ構造は、*ボロノイ・セル構造* と呼ばれています。ボロノイ分割 によってできるのは、各セルの内部が、他のどのグリッド点よりも特定のグリッド点に近いポイントばかりから構成されるセル構造です。ボロノイ・セル構造中のセルは、*凸包* になります。ポイントの集合の凸包は、そのポイント集合を含む最小の凸集合 (あるいは、ポイント集合の「外部」を定義する最小のポリゴン) です。ボロノイ・セル構造 は、不規則な形状のポリゴンになりやすい傾向にあります。セルの数や配置は、空間データの密度や配置に合うようチューニングできます。

例えば、ボロノイ・セル構造を利用すれば、地球を、人口を基準をポリゴンに分割するといったことができます。人口 (およびデータ) の密度が高い部分では、ポリゴンを小さくするといったことをするのです。逆に、人口密度が低い部分ではポリゴンを大きくします。

149 ページの図 21 は、世界の人口密度を基準にしたボロノイ構造の例です。Geodetic Data Management Feature はこのセル構造を、空間計算で利用します。

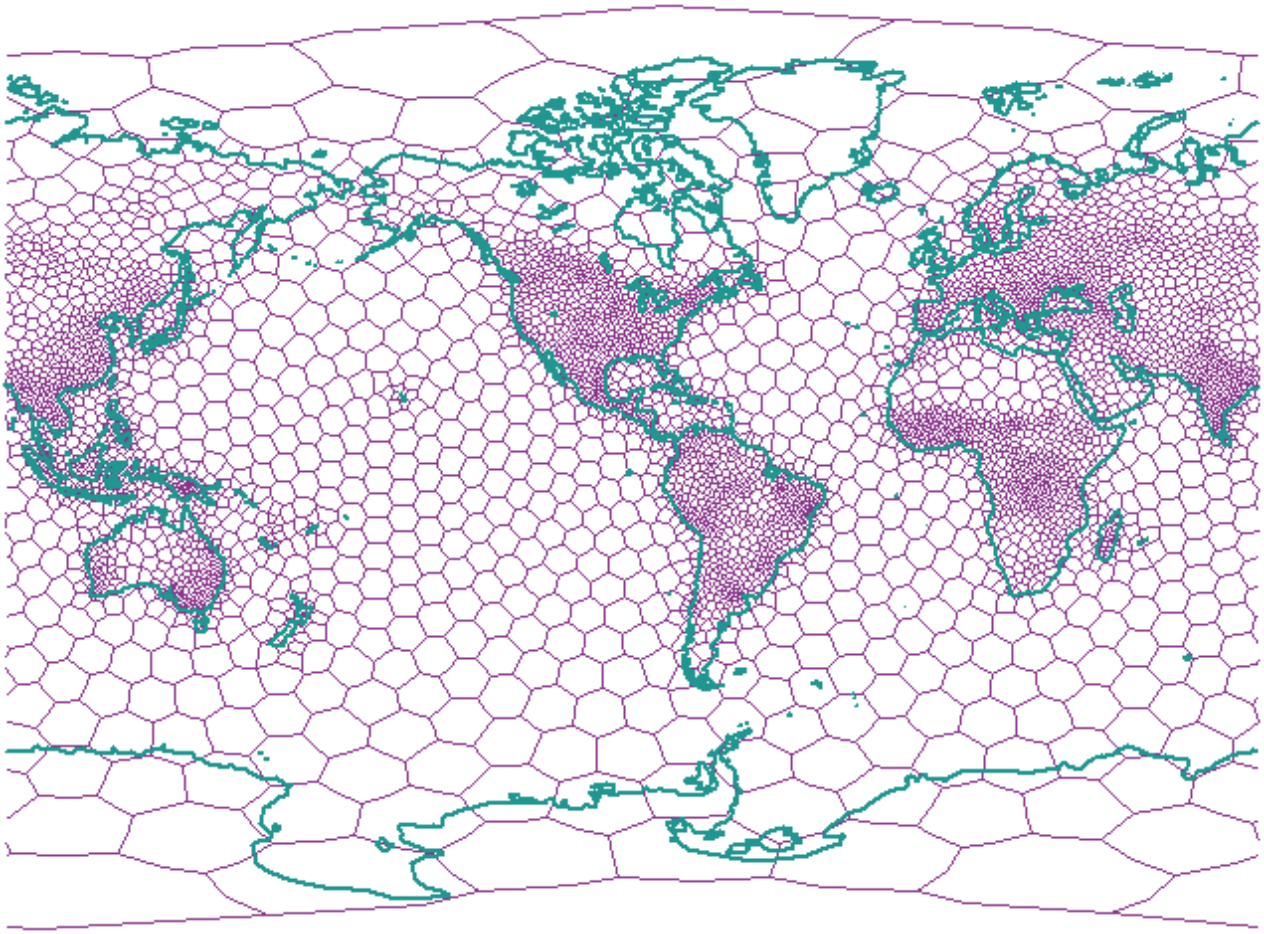


図 21. 世界の人口密度を基準にしたボロノイ構造の例

代替ボロノイ・セル構造を選択する際の考慮事項

測地形状に対する操作では、常に、世界の人口密度を基準にしたボロノイ・セル構造に対応するボロノイ ID 「1」を使用します。 索引作成時、データが地球上の一定の領域に集中している場合 (特定の国の街路に関するデータが対象になる場合など) には、データが位置する領域でセルが小さくなる (解像度はセル・サイズに反比例するため) 代替ボロノイ・セル構造を選択できます。 DB2® Geodetic Data Management Feature は、個々のユーザーが使用するデータに適合するものを選択できるよう、索引付けのボロノイ・セル構造を多数提供しています。

制約事項:
代替のボロノイ・セル構造は測地ボロノイ・インデックスを作成するときにのみ選択できます。

dodeca04 構造 (ボロノイ ID 12) は、衛星画像など地球表面全体に均一に分散するデータに最も適合します。セルのサイズはすべてほぼ均一で、解像度は最も低い部分で、約 10 cm です。データやアプリケーションが以下のいずれかの条件に当ては

まる場合には、世界の人口密度を基準にしたデフォルトの (ボロノイ ID 1 の) セル構造、あるいは dodeca04 以外のボロノイ・セル構造を使用することを検討します。

高い解像度が必要

互いの間の距離が 10 cm 未満のオブジェクトが交差しているかを確認する必要がある場合には、該当データが位置している領域についてはセル・サイズの小さいボロノイ・セル構造を使用する必要があります。解像度は、セル・サイズに反比例します。

ポリゴンの頂点の数が多い

データが、比較的頂点の数が多く、面積が比較的小さいポリゴンで構成される場合には、ボロノイ・セル構造を、該当領域中に多くのセルを持つものに変える必要があります。多くのポリゴンの頂点数が 50 以下であれば、通常、その必要はありません。データ・セット中に頂点数の多いポリゴンがあっても、それが大陸レベルのサイズのものばかりである場合には、ボロノイ・セル構造の変更は通常必要ありません。

3000 もの頂点を持ち、サイズが米国ほどのポリゴンが多数ある、という場合、特に、アプリケーションがポリゴンとポリゴンの交差に関連する照会を数多く実行する場合には、 ボロノイ・セル構造に切り替えることで、かなり照会のパフォーマンスを向上できる可能性があります。

データの密度が非常に高い

データが小さい領域に集中している場合 (例えば、1 平方キロメートルの中に数百のオブジェクトが集中しているような場合)、セル密度がデータの密度に合ったボロノイ・セル構造を利用することで照会のパフォーマンスを向上できる可能性があります。

測地ボロノイ・インデックスの作成

前提条件

測地ボロノイ・インデックスを作成する際には、空間グリッド・インデックスの作成に使用したのと同じ権限、特権を持つユーザー ID が必要です。

制約事項:
測地ボロノイ・インデックスを作成するときも、CREATE INDEX ステートメントを使用して索引を作成する際の制約が当てはまります。すなわち、索引を作成する列は、ビュー列またはニックネーム列ではなく、基本表の列でなければなりません。DB2 データベース・システムは、処理中に別名を解決します。

DB2 Geodetic Data Management Feature は、測地データを含む列についての索引の作成を可能にする新しい空間アクセス方法を提供しています。索引を使用すれば、照会の実行は速くなります。

測地ボロノイ・インデックスは、以下のいずれかの方法によって作成できます。

- DB2 コントロール・センターの「索引の作成」ウィンドウを使用する。
- EXTEND USING 節で db2gse.spatial_index 拡張子を付けて SQL CREATE INDEX ステートメントを使用する。

DB2 コントロール・センターまたはコマンド行プロセッサーを使用して測地ボロノイ・インデックスを作成します。

- コントロール・センターを使用して測地ボロノイ・インデックスを作成するには、測地ボロノイ・インデックスを作成する空間列を持つ表を右クリックし、メニューから「**Spatial Extender**」→「**空間インデックス (Spatial Indexes)**」をクリックします。「空間インデックス」ウィンドウが開きます。プロンプトに従ってこのタスクを完了します。
- SQL CREATE INDEX ステートメントを使用して測地ボロノイ・インデックスを作成するには、EXTEND USING 節と db2gse.spatial_index グリッド索引拡張子を使用して CREATE INDEX ステートメントを発行します。

以下の例では、CREATE INDEX ステートメントは、CUSTOMERS 表の LOCATION 空間列に STORESX1 測地索引を作成します。

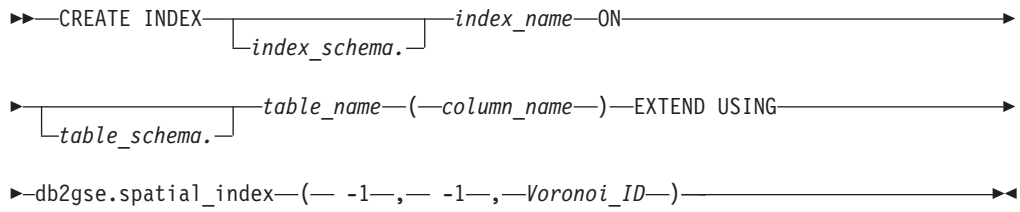
```
CREATE INDEX storesx1
   ON customers (location)
   EXTEND USING db2gse.spatial_index (-1, -1, 1)
```

測地ボロノイ・インデックスの場合、USING db2gse.spatial_index 節の最初の 2 つのパラメーターに値「–1」を指定する必要があります。

測地ボロノイ・インデックスを作成するための CREATE INDEX ステートメント

測地ボロノイ・インデックスを作成する場合は、CREATE INDEX ステートメントに EXTEND USING 節を指定して使用してください。

構文:



それぞれの意味を説明します。

index_schema
作成する索引が属するスキーマの名前。名前を指定しないと、DB2 データベース・システムは CURRENT SCHEMA 特殊レジスターに保管されているスキーマ名を使用します。

index_name
作成する測地索引の非修飾名。

table_schema
column_name を含んでいる表が属するスキーマの名前。名前を指定しないと、DB2 データベース・システムは CURRENT SCHEMA 特殊レジスターに保管されているスキーマ名を使用します。

table_name
column_name を含んでいる表の非修飾名。

column_name
測地ボロノイ・インデックスを作成する空間列の名前。

Voronoi_ID
ボロノイ・セル構造のID (整数)。使用できるボロノイ・セル構造は 14 個あります。ボロノイ ID 1 は、DB2 Geodetic Data Management Feature によるすべての空間操作にも使用される世界の人口密度を基礎としたボロノイ・セル構造を指定します。

例

以下に例として示した CREATE INDEX ステートメントでは、CUSTOMERS 表の LOCATION 空間列についての STORESX1 測地索引が作成されます。

```
CREATE INDEX storesx1
   ON customers (location)
   EXTEND USING db2gse.spatial_index (-1, -1, 1)
```

オプティマイザーは、Where 節中に以下の関数を含む照会のすべてについてボロノイ索引の使用を検討します。

- ST_Contains
- ST_Distance
- ST_Intersects
- ST_MBRIntersects
- ST_EnvIntersects
- EnvelopesIntersect
- ST_Within

以下のステートメントは、ボロノイ索引の使用例です。まず、CUSTOMER 表にデータを挿入します。最初の INSERT ステートメントのように直接値を入力することができます。

```
INSERT INTO customer
(id, last_name, first_name, address, city, state, zip,
location)
VALUES
('123-456789', 'Duck', 'Donald',
'123 Mallard Way', 'Wetland Marsh', 'ND', '55555-5555',
db2gse.ST_GeomFromWKT('POINT(123.123, 45.67)', 2000000000))
```

その代わりに、次の照会のように、アプリケーション中の変数を使用して値を表に挿入することもできます。

```
INSERT INTO customer
(id, last_name, first_name,
address, city, state, zip,
location)
VALUES
(:mid, :mlast, :mfirst,
:maddress, :mcity, :mstate, :mzip,
db2gse.ST_GeomFromWKB(:mlocation))
```

次の UPDATE ステートメントでは、挿入したデータに変更を加えます。WHERE 節で ST_Contains、 ST_Distance、 ST_Intersects、 ST_MBRIntersects、 ST_EnvIntersects、EnvelopesIntersect、 ST_Within といった関数を使用していないので、STORESX1 索引は使用されません。

```
UPDATE customer
SET location = db2gse.ST_GeomFromWKT('POINT(123.123, 45.67)',
2000000000)
WHERE id = '123-456789';
```

以下の DELETE ステートメントでは、ST_Within function 関数、ST_Intersects 関数が Where 節で使用されているので、索引を使用することでパフォーマンスが向上するとオプティマイザーが判断した場合には、STORESX1 索引が使用されます。

```
DELETE FROM customers
WHERE db2gse.ST_Within(location, :BayArea) = 1;
```

```
DELETE FROM customers
WHERE db2gse.ST_Intersects(c.location, :BayArea) = 1
```

以下の 2 つの SELECT ステートメントでも、STORESX1 索引が使用される可能性があります。

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

DB2 Geodetic Data Management Feature で提供されるボロノイ・セル構造

ボロノイ・セル構造はいずれも地球全体を網羅しています。後の図では、ボロノイ・セル構造の中でも、セルの高密度な領域のみを示してあります。ボロノイ・セル構造を選択する際には、図に示された以外の領域ではセルが大きくなり、それに伴い解像度が下がる点に注意してください。使用するデータが密度の薄い領域に位置する場合には、照会のパフォーマンスが下がる恐れがあります。

以下の表は、DB2 Geodetic Data Management Feature が提供するボロノイ・セル構造のリストです。これらのボロノイ・セル構造は、 Geodyssey Ltd. によって提供されているものです。

表 23. ボロノイ・セル構造

説明	ボロノイ ID	対応する図
世界の人口密度を基準にしたもの	1	154 ページの図 22
アメリカ合衆国	2	155 ページの図 23
カナダ	3	156 ページの図 24
インド	4	157 ページの図 25
日本	5	158 ページの図 26
アフリカ	6	159 ページの図 27
オーストラリア	7	160 ページの図 28
ヨーロッパ	8	161 ページの図 29
北アメリカ	9	162 ページの図 30
南アメリカ	10	163 ページの図 31
地中海	11	164 ページの図 32
世界全体に均一にデータが分散、解像度は中程度 (dodeca04)	12	165 ページの図 33

表 23. ボロノイ・セル構造 (続き)

説明	ボロノイ ID	対応する図
世界の工業生産高を基準にしたもの (G7 諸国)	13	166 ページの図 34
世界全体に均一にデータが分散、解像度は低い (isotype)	14	167 ページの図 35

世界の人口密度を基準にしたセル構造 (ボロノイ ID: 1)

ボロノイ・セルは、世界の人口密度に基づいて地球を細分化します。

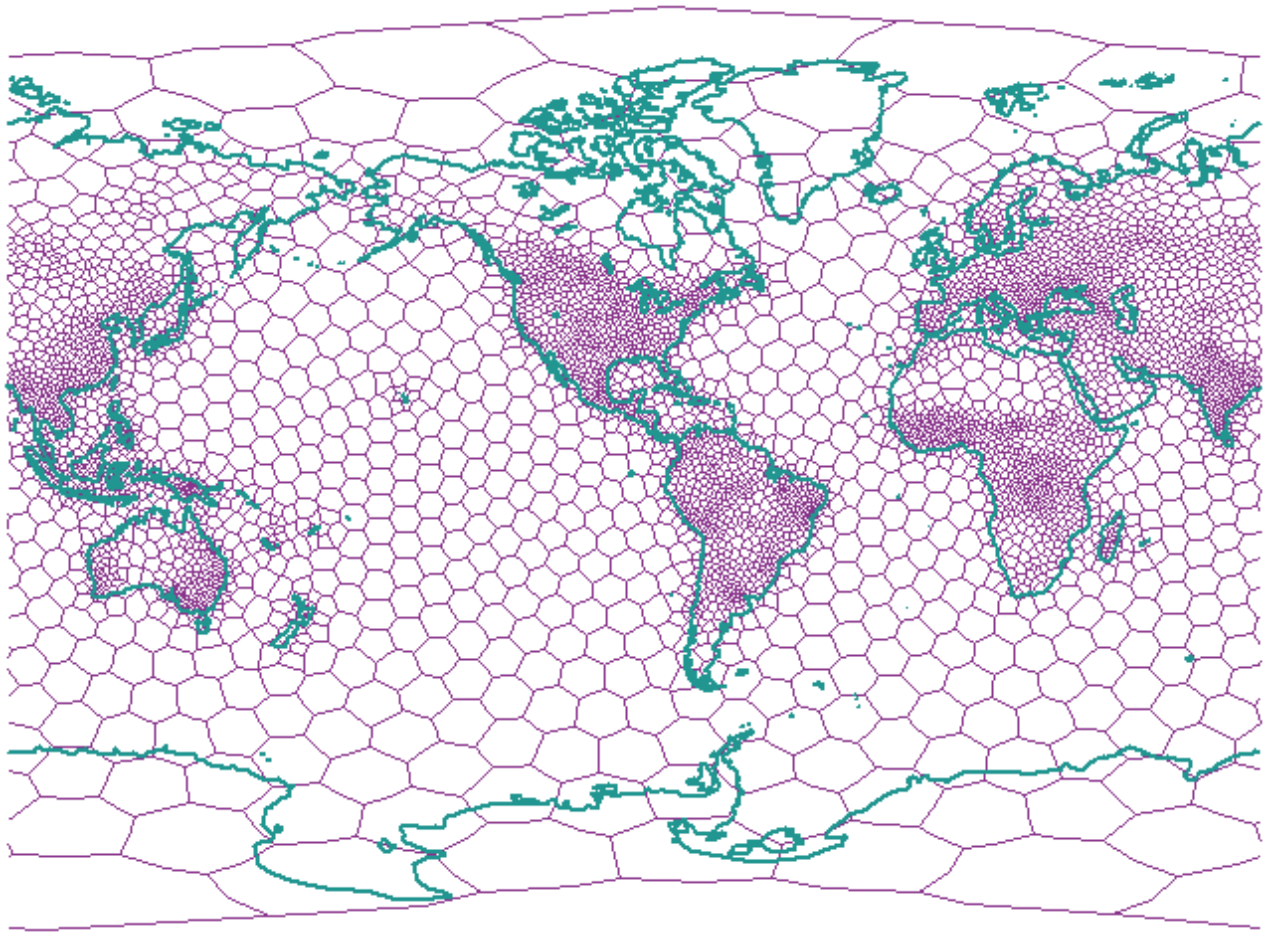


図 22. 世界全体 (の人口密度) を対象とするボロノイ・セル構造

米国 (ボロノイ ID: 2)

ボロノイ・セルは、人口密度に基づいてアメリカ合衆国を細分化します。

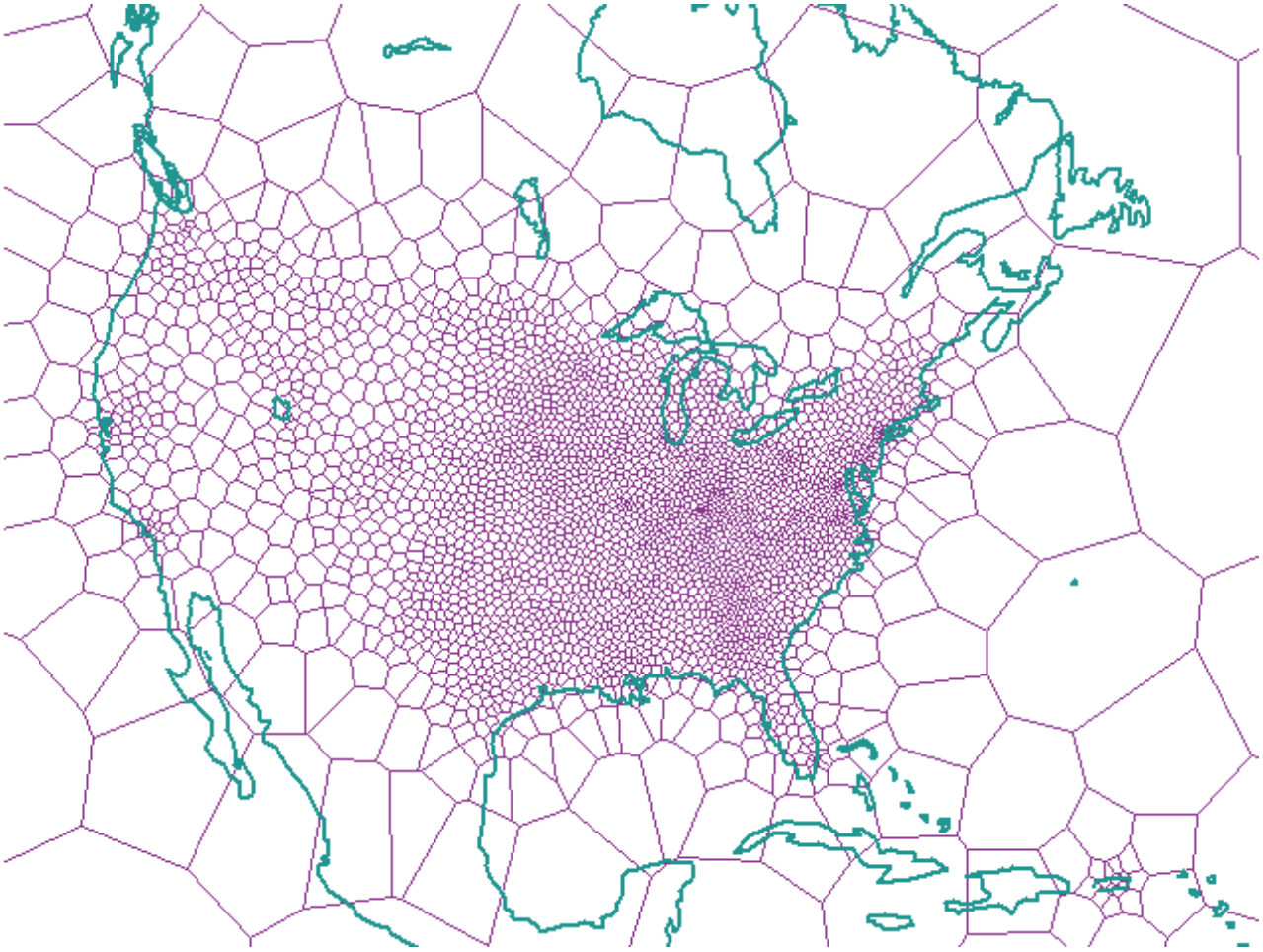


図 23. 米国を対象とするボロノイ・セル構造

カナダ (ボロノイ ID: 3)

ボロノイ・セルは、人口密度に基づいてカナダを細分化します。

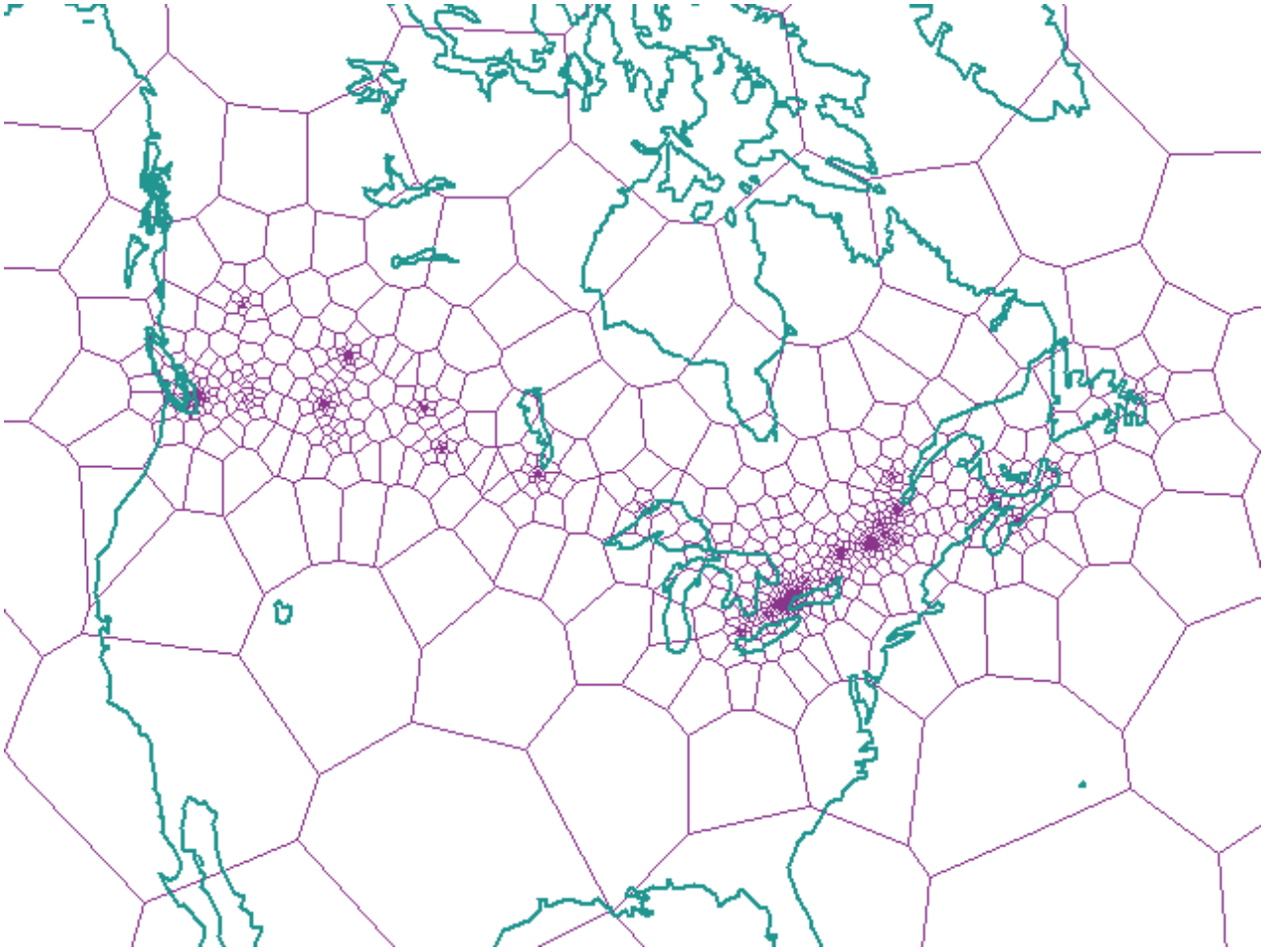


図 24. カナダを対象とするボロノイ・セル構造

インド (ボロノイ ID: 4)

ボロノイ・セルは、人口密度に基づいてインドを細分化します。

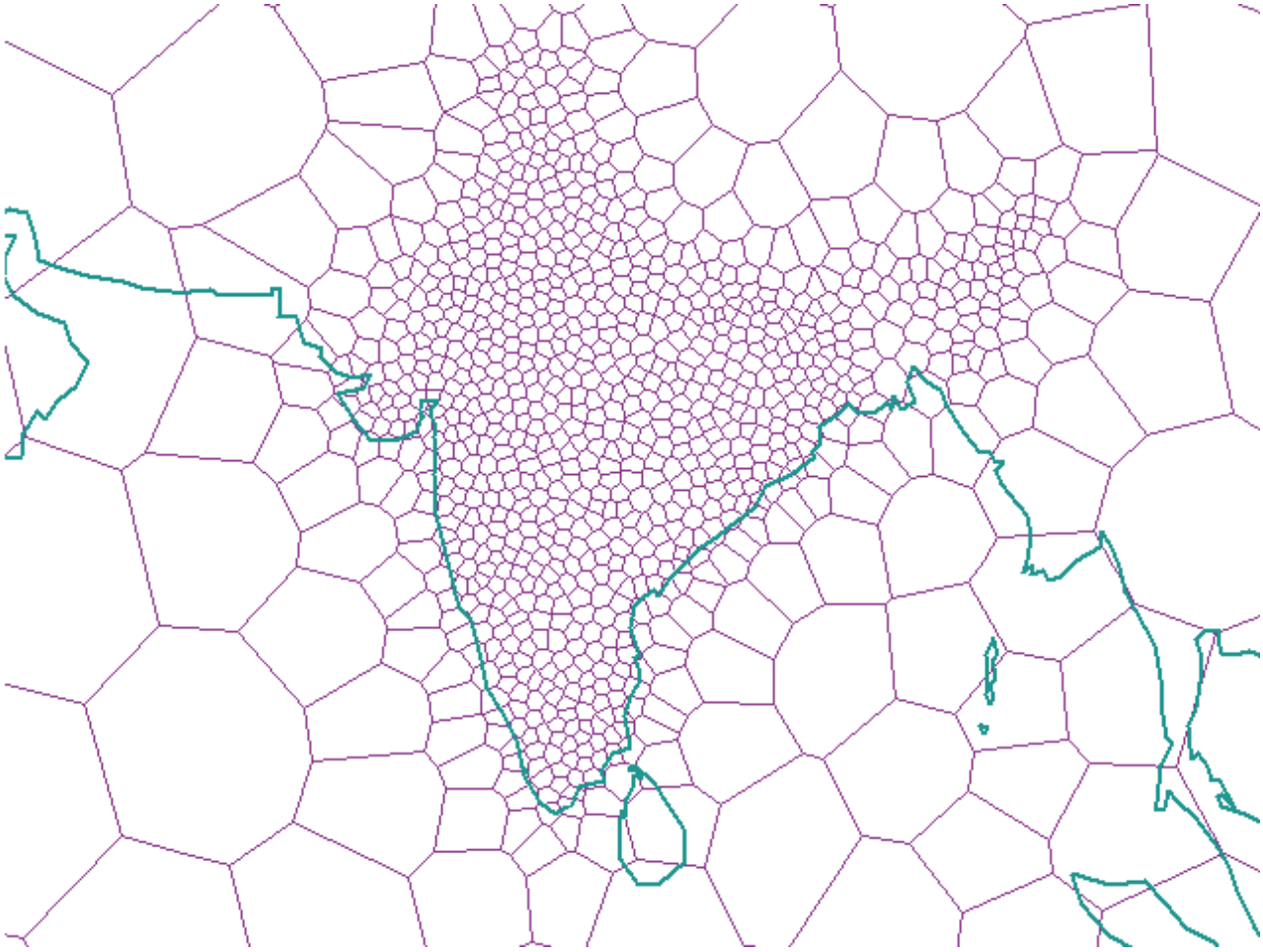


図 25. インドを対象とするボロノイ・セル構造

日本 (ボロノイ ID: 5)

ボロノイ・セルは、人口密度に基づいて日本を細分化します。

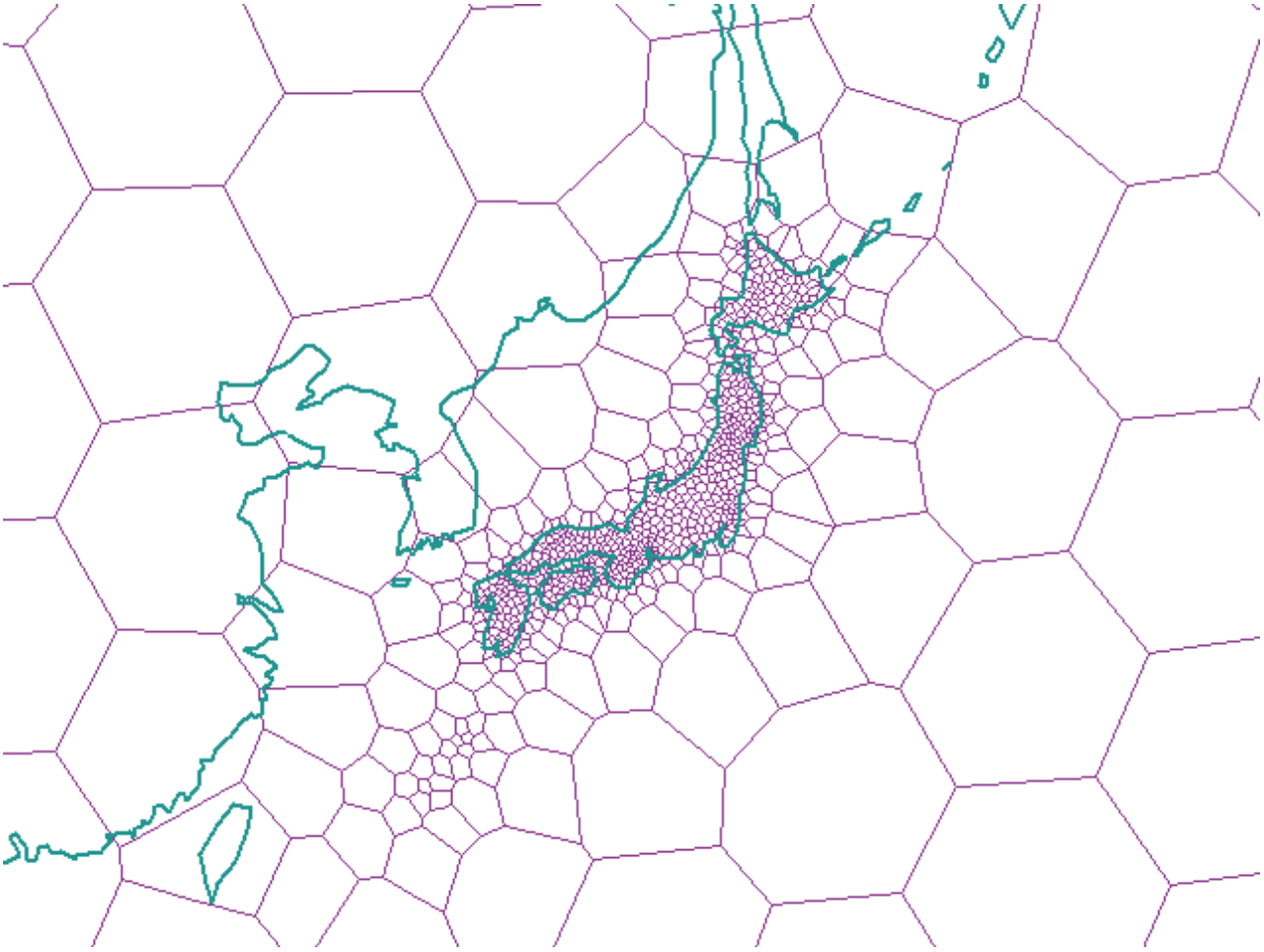


図 26. 日本を対象とするボロノイ・セル構造

アフリカ (ボロノイ ID: 6)

ボロノイ・セルは、人口密度に基づいてアフリカを細分化します。

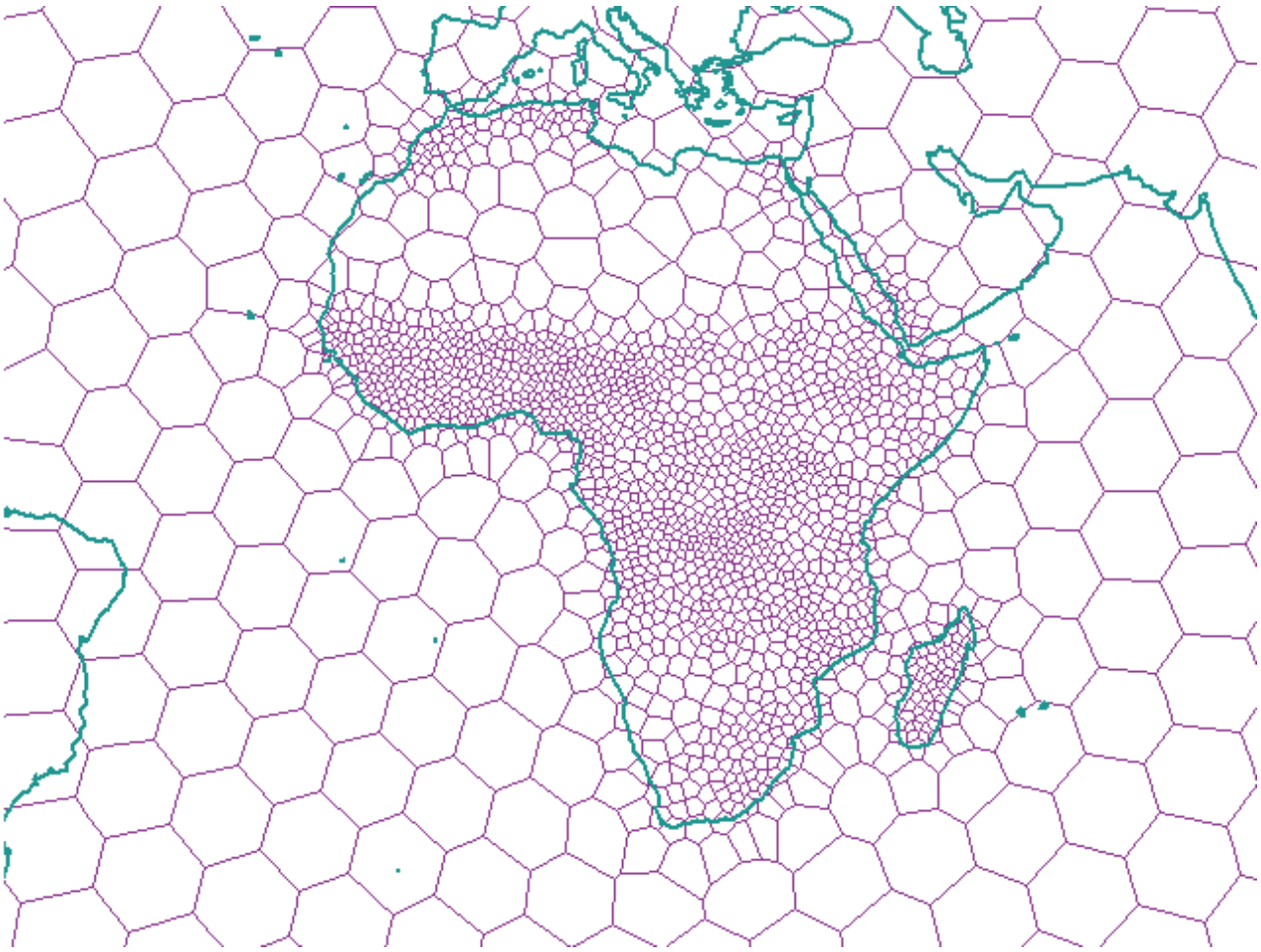


図 27. アフリカを対象とするボロノイ・セル構造

オーストラリア (ボロノイ ID: 7)

ボロノイ・セルは、人口密度に基づいてオーストラリアを細分化します。

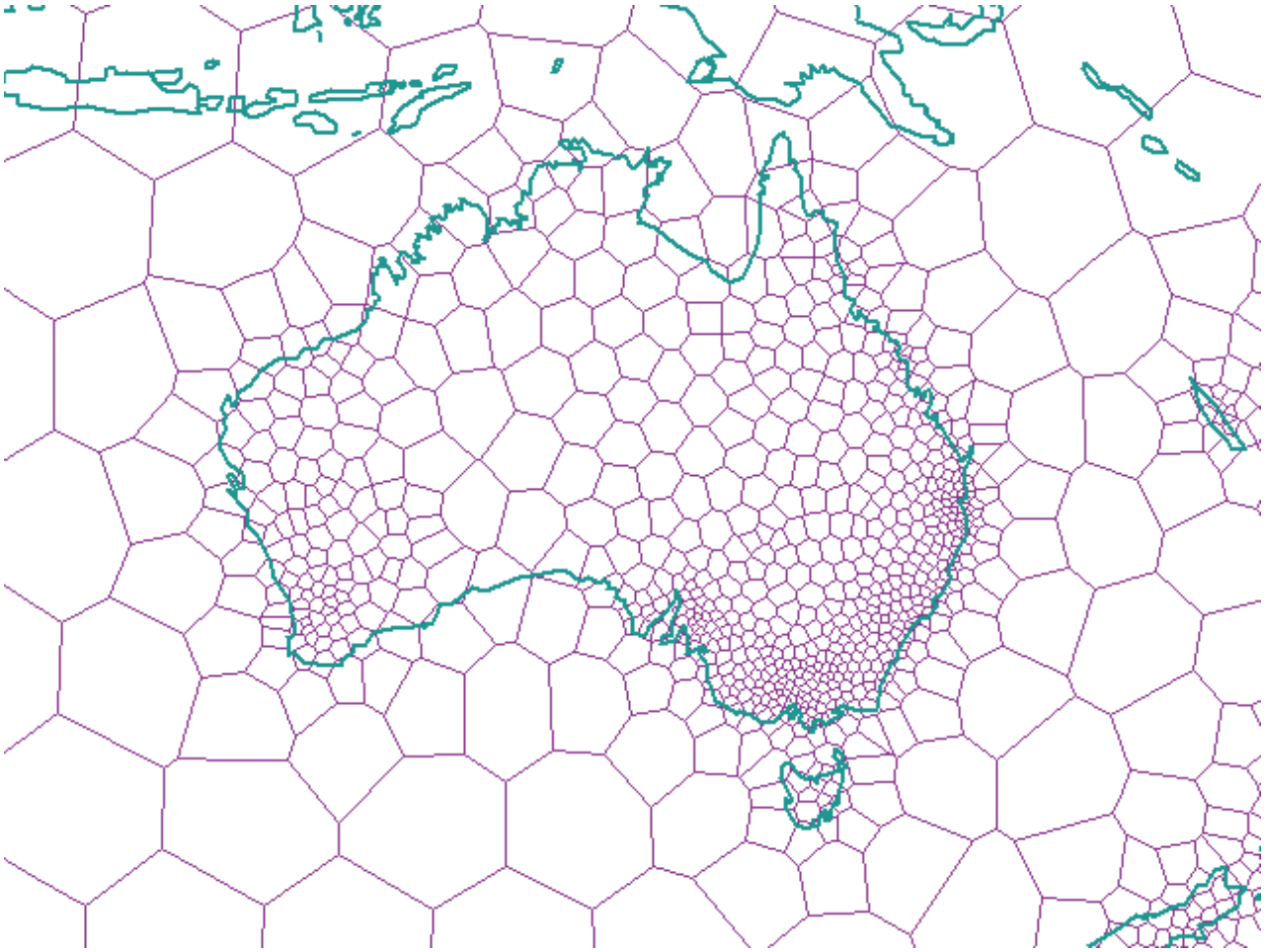


図 28. オーストラリアを対象とするボロノイ・セル構造

ヨーロッパ (ボロノイ ID: 8)

ボロノイ・セルは、人口密度に基づいてヨーロッパを細分化します。

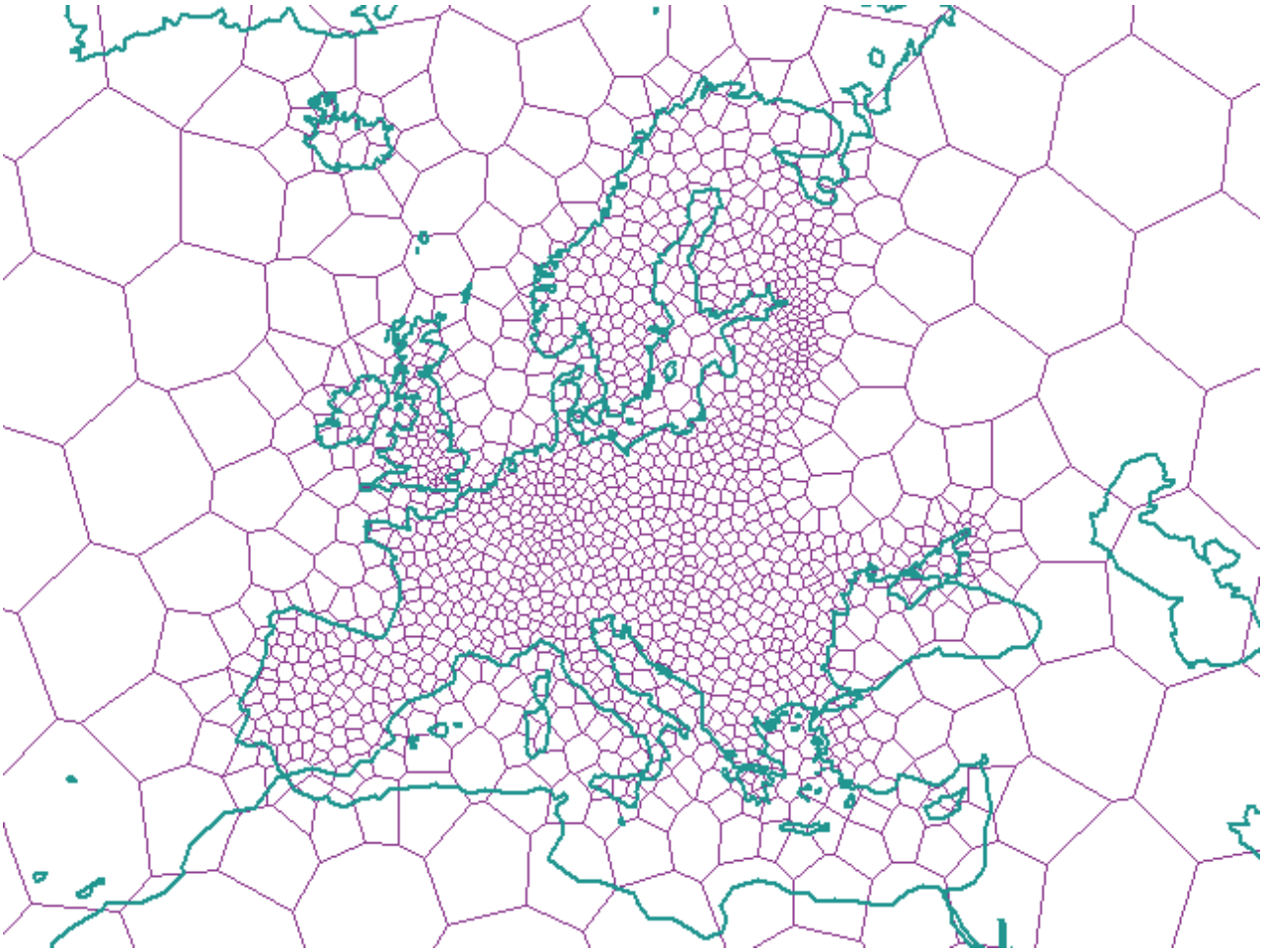


図 29. ヨーロッパを対象とするボロノイ・セル構造

北アメリカ (ボロノイ ID: 9)

ボロノイ・セルは、人口密度に基づいて北アメリカを細分化します。

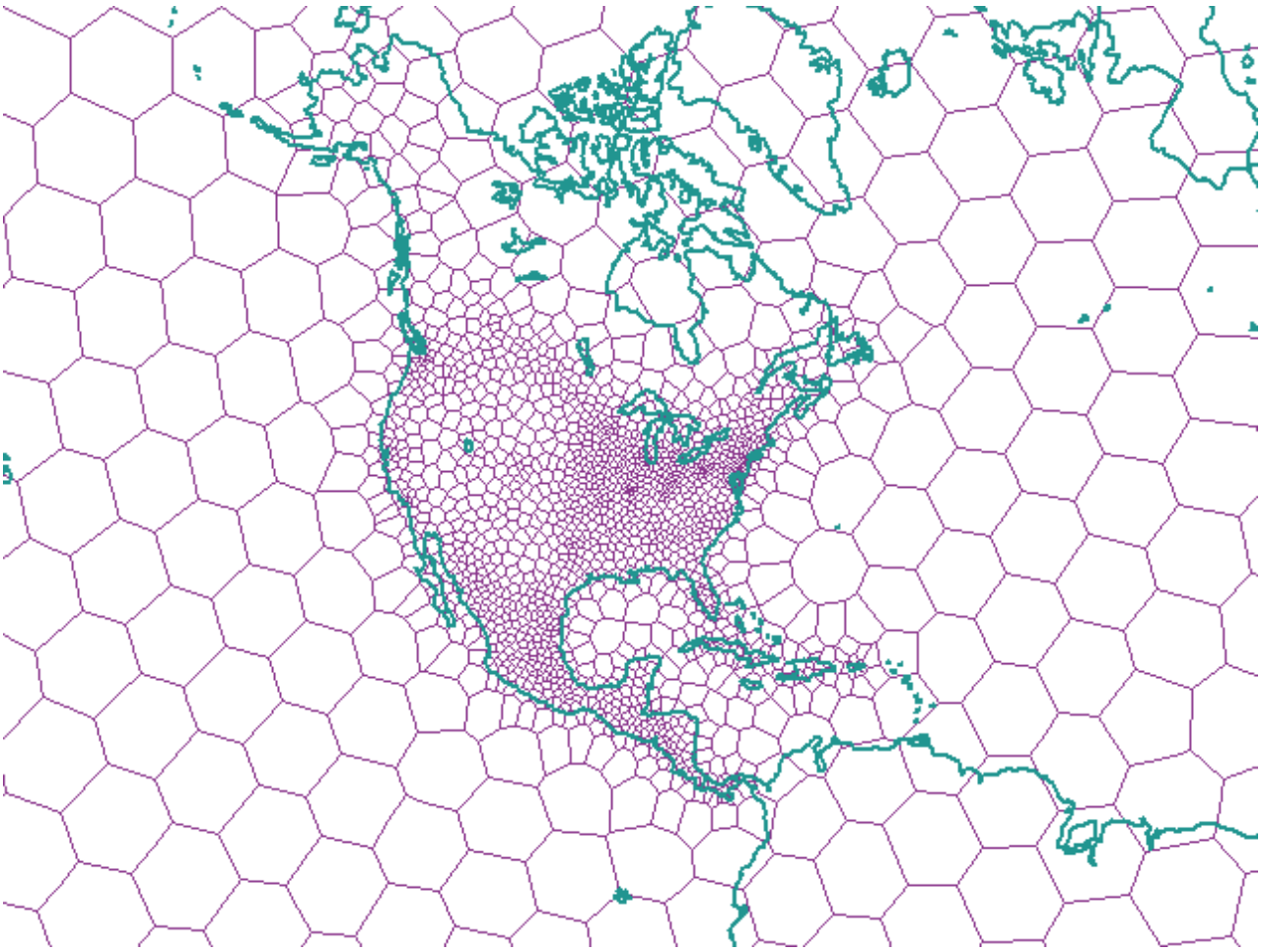


図 30. 北アメリカを対象とするボロノイ・セル構造

南アメリカ (ボロノイ ID: 10)

ボロノイ・セルは、人口密度に基づいて南アメリカを細分化します。

図 31. 南アメリカを対象とするボロノイ・セル構造

地中海 (ボロノイ ID: 11)

ボロノイ・セルは、人口密度に基づいて地中海地域を細分化します。

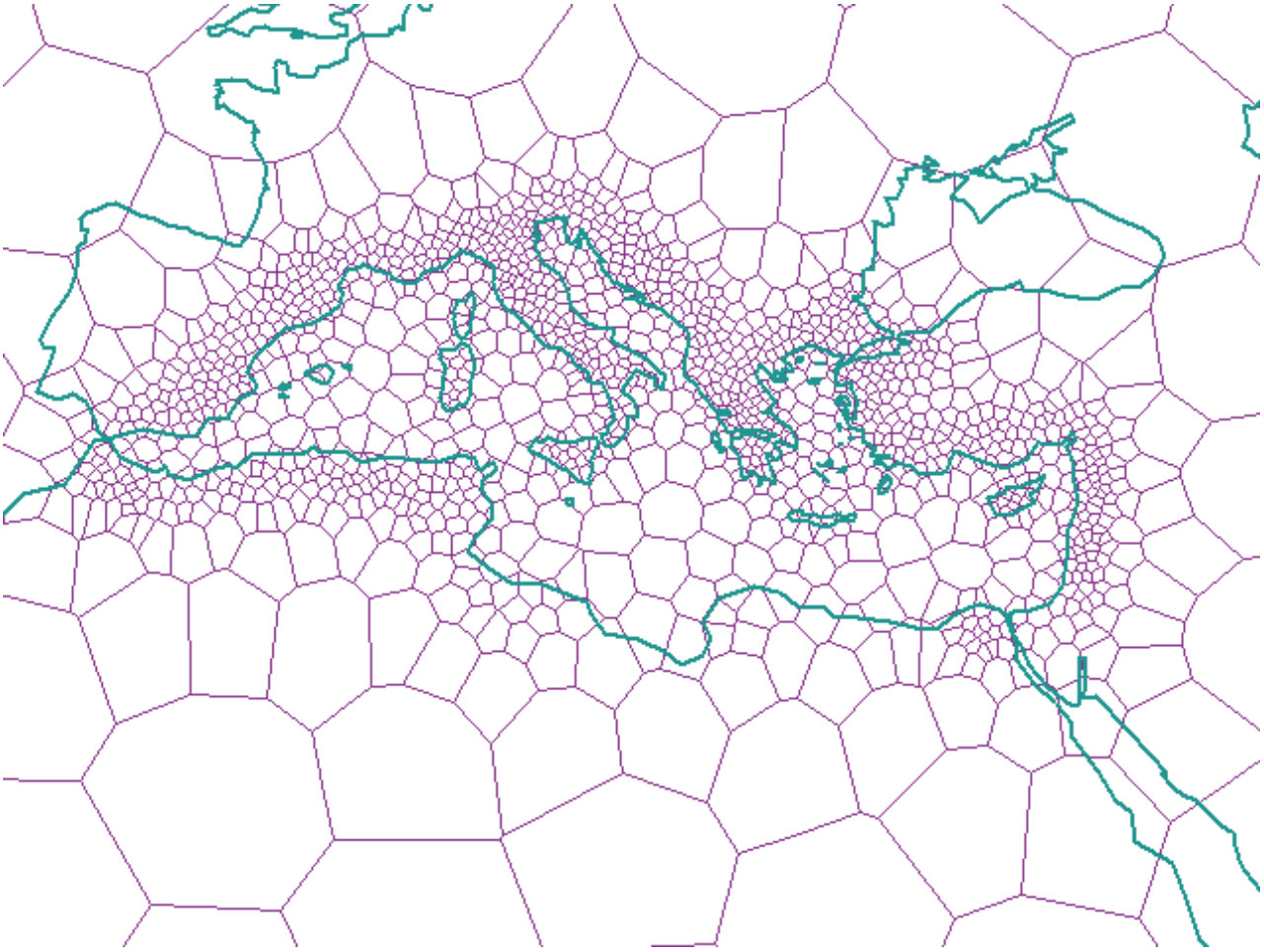


図 32. 地中海地域を対象とするボロノイ・セル構造

世界全体に均一にデータが分散、解像度は中程度 – dodeca04 (ボロノイ ID: 12)

ボロノイ・セルは、均一にデータが分散されている世界を中解像度で細分化します。

図 33. 世界全体を対象とするボロノイ・セル構造 (dodeca04)

世界の工業国を対象 – G7 諸国 (ボロノイ ID: 13)

ボロノイ・セルは、各国の工業生産高に基づいて世界を細分化します。

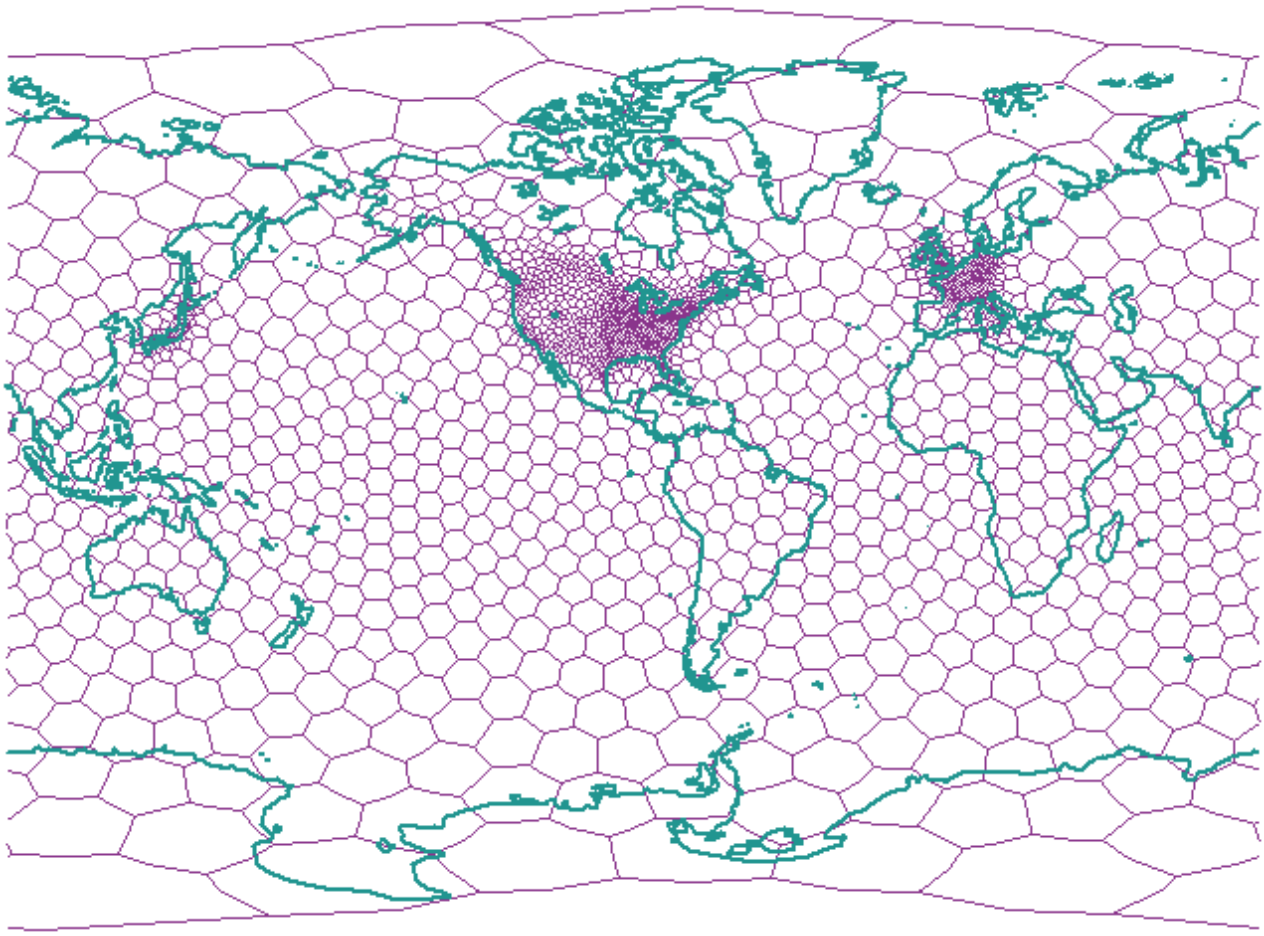


図 34. G7 諸国を対象とするボロノイ・セル構造

世界全体に均一にデータが分散、解像度は低い – isotype (ボロノイ ID: 14)

ボロノイ・セルは、均一にデータが分散されている世界を低解像度で細分化します。

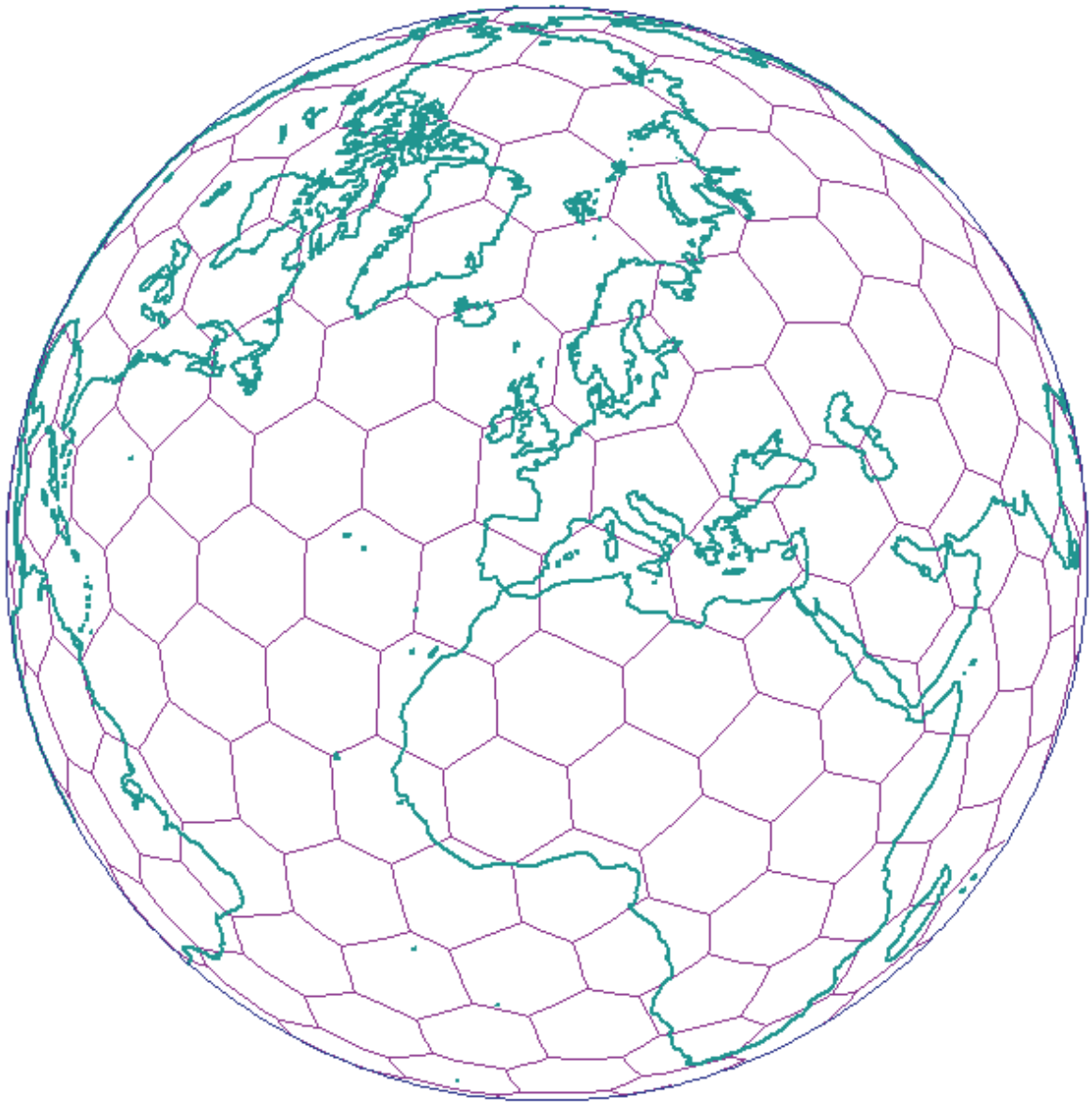


図 35. 世界全体を対象とするボロノイ・セル構造 (isotype)

第 19 章 測地データを使用した場合と空間データを使用した場合の違い

この章では、測地データを使用した場合と空間データを使用した場合の以下の違いについて説明しています。

- ST_Geometry データ・タイプの x、y 属性の最小値、最大値
- 平面地球を使用した場合と球体地球を使用した場合の違い
- DB2 Geodetic Data Management Feature によってサポートされる空間処理関数、および関数の動作の違い
- DB2 Geodetic Data Management Feature によってサポートされるストアード・プロシージャーおよびカタログ・ビュー
- その他の測地参照系 (測地系) および測地楕円

x、y 属性の最小値、最大値

DB2® Geodetic Data Management Feature は、データを測地ボロノイ・インデックス用にセル構造に編成するために、最小外接長方形の代わりに、最小外接円 (MBC) を使用します。

測地形状の場合、MBC とは、形状を囲む円のことで、最小、最大 の x、y は以下のような内部値を持ちます。

xmin 外接円の中心からコサイン方向の *i* 項。

xmax 外接円の中心からコサイン方向の *j* 項。

ymin 外接円の中心からコサイン方向の *k* 項。

ymax 外接円の *arc_radius*。

測地形状の場合、ST_MinX、ST_MaxX、ST_MinY、 ST_MaxY などの関数は、MBC に沿ったポイントを表示します。こうした関数によって作られる経度、緯度の値は、空間形状のものに似ていますが、関数の実行結果は、測地形状の場合、以下のように異なっています。

- MBC が日付変更線をまたぐ場合は、ST_MinX 値は ST_MaxX 値より大きくなる。例えば、MBC の中心が日付変更線上にあり、MBC の半径が 5 度であれば、ST_MinX の値は 175、ST_MaxX の値は –175 になります。
- MBC が北極点あるいは南極点を含む場合、ST_MinX の値は –180、ST_MaxX は 180 になる。
- MBC が北極点を含む場合、ST_MaxY の値は 90 になる。
- MBC が南極点を含む場合、ST_MinY の値は –90 になる。

© Copyright IBM Corp. 1998, 2009

平面地球を使用した場合と球体地球を使用した場合の違い

DB2 Spatial Extender と DB2 Geodetic Data Management Feature では、中核となる技術が異なります。

- Spatial Extender では、投影座標を基礎とした平面の地図を使用します。しかし、地図投影では、決して、地球を正確に表現することはできません。地球には、「端」というものは存在しないのですが、地図には必ず「端」があるからです。
- Geodetic Data Management Feature では、楕円をモデルとして使用して、地球を継ぎ目のない球体として扱い、極地で歪曲したり、180 度子午線で途切れたりしないようにします。

このセクションでは、「平面地球」という言葉を、投影による地球全体の表現という意味に使用します。「球体地球」という言葉は、地球のモデルとして楕円を利用した参照系という意味に使用します。

使用する技術が異なれば、様々な状況での形状の扱い方も変わってきます。違いが生じるのは、例えば、以下に示すようなことに関してです。

- 180 度子午線をまたぐ直線セグメント (および測定距離)。
- 180 度子午線をまたぐポリゴン。
- 180 度子午線をまたぐ最小外接長方形。
- 極点を囲むポリゴン
- 半球、赤道地帯、地球全体を表すポリゴン。

Geodetic Data Management Feature の特長としてまずあげられるのは、180 度子午線にまたがる形状や、極地に近い形状など、Spatial Extender の平面地球による表現では限界がある形状を問題なく表現できるという点です。

180 度子午線をまたぐ直線セグメント

球体地球表現は平面地球投影よりも短い経路を提供します。

171 ページの図 36 は、Spatial Extender と Geodetic Data Management Feature の、180 度子午線にまたがる直線セグメントの扱い方の違いを示したものです。ここでの例では、直線セグメントは、アンカレッジと東京の間の距離を測定するために使用するものになっています。 Geodetic Data Management Feature では、測地線、つまり楕円上の 2 点間の最短経路に沿って、2 点間の距離を測定します。この場合の 2 点は、地球上のいずれの場所でも構いません。球体による地球表現を利用しているため、2 点がアンカレッジと東京の場合でも、Geodetic Data Management Feature はアンカレッジから西回りで東京に到達する直線セグメントを正しく選択できます。一方の Spatial Extender は平面地図投影を利用するため、アンカレッジから東京に到達する西回りの経路を発見できず、はるかに遠い東回りの直線セグメントを選択してしまいます。平面地図投影では、–180 度子午線は左端に、180 度子午線は右端に位置します。

Spatial Extender を利用してただし結果を得るには、以下のいずれかの方法を採る必要があります。

- 直線セグメントを、180 度子午線の東と西の 2 つの直線セグメントに分割する。
- 180 度子午線が端にならないようデータの投影をし直す。

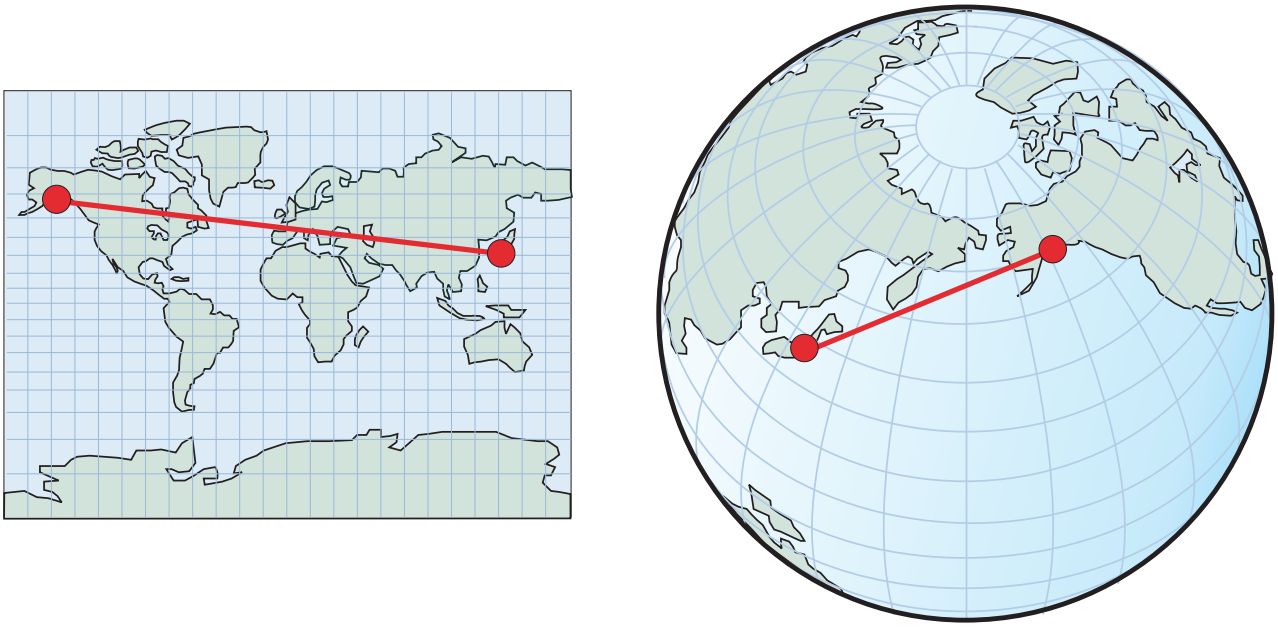


図 36. 180 度子午線をまたぐ直線

180 度子午線をまたぐポリゴン

180 度子午線をまたぐポリゴンを扱うには、平面地球表現 (Spatial Extender) の場合、ポリゴンを 2 つの部分に分割することが必要です。

次の例は、180 度子午線の東の部分のポリゴンと、180 度子午線の西の部分のポリゴンを示しています。

```
  MULTIPOLYGON(
((-180 30, -165 30, -165 40, -180 40, -180 30)),
((180 30, 180 40, 165 40, 165 30, 180 30)))
```

172 ページの図 37 のように、球体地球表現 (Geodetic Data Management Feature) では、そのような分割は不要で、1 つのポリゴンを手を加えずに使用できます。

```
  POLYGON((165 30, -165 30, -165 40, 165 40, 165 30))
```

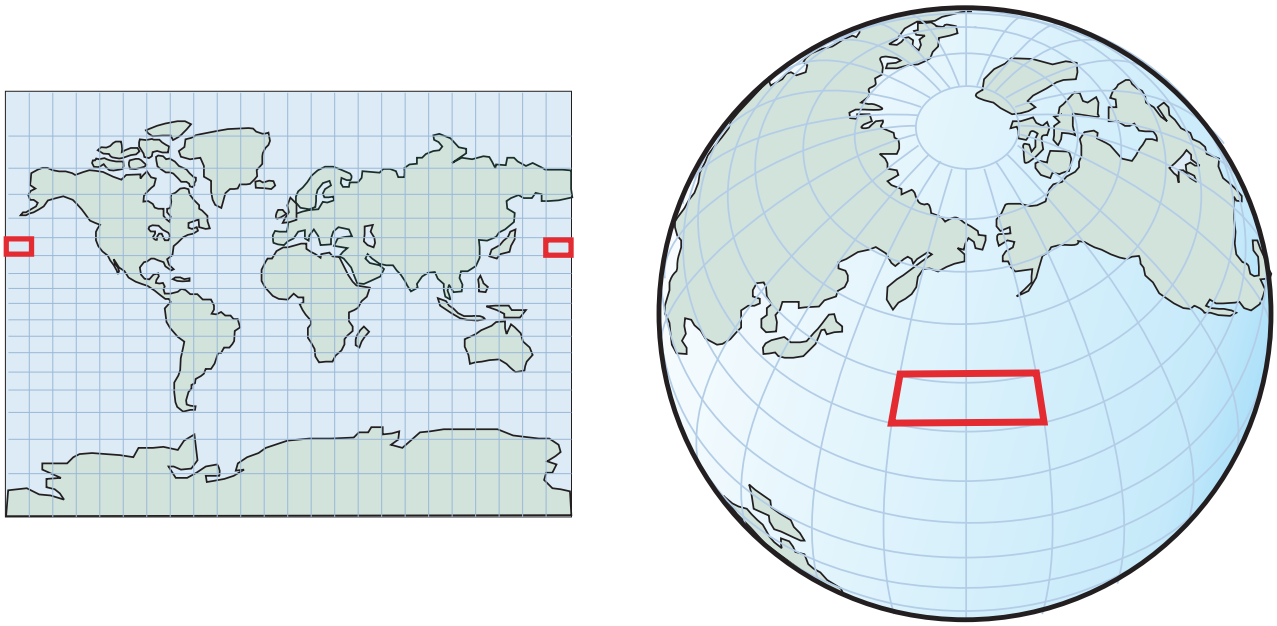


図 37. 180 度子午線をまたぐポリゴン — ポリゴンの分割

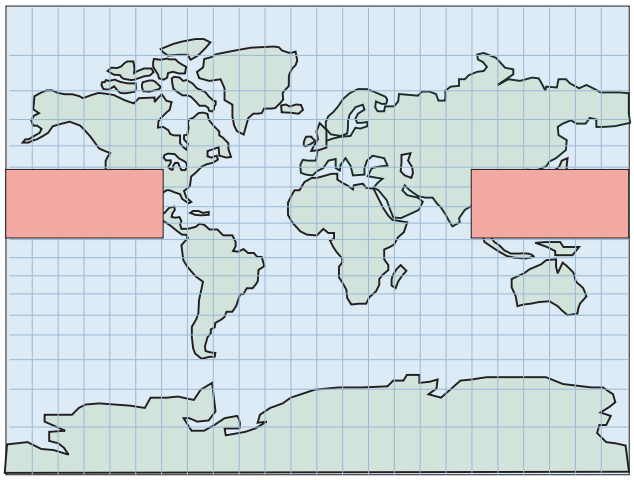
Spatial Extender を使用しているにもかかわらずポリゴンを分割しなかった場合には、 173 ページの図 38 に示すような別の領域を定義するため、ポリゴンの頂点の再配列が行われることになります。 173 ページの図 38 の上側は、180 度子午線をまたぐポリゴンの正しい配置を示しています。

```
POLYGON((90 0, -90 0, -90 40, 90 40, 90 0))
```

173 ページの図 38 の下側は、頂点の再配列によって結果的に 180 度子午線をまたがず、0 度子午線をまたぐようになった、誤ったポリゴンを示しています。

```
POLYGON((-90 0, 90 0, 90 40, -90 40, -90 0))
```

第 180 子午線をまたぐポリゴン
Polygon ((90 0, -90 0, -90 40, 90 40)) が必要



しかし、Spatial Extender は、頂点を再配列してしまうので、
その結果、違った領域を定義するポリゴン
Polygon ((-900, 90 0, 90 40, -90 40)) ができてしまう

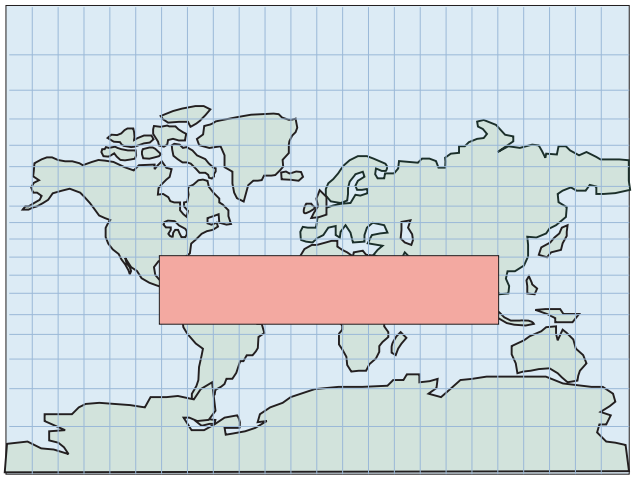


図 38. 180 度子午線をまたぐポリゴン—頂点の再配列

再配列によってできるポリゴンが占める領域は、 174 ページの図 39 に示すように、ユーザーの意図する領域ではなく、それを補完するようなものになります。 先述の直線セグメントと同様、180 度子午線が端にならないようにデータの投影をし直すという方法でこの問題に対処することも可能です。

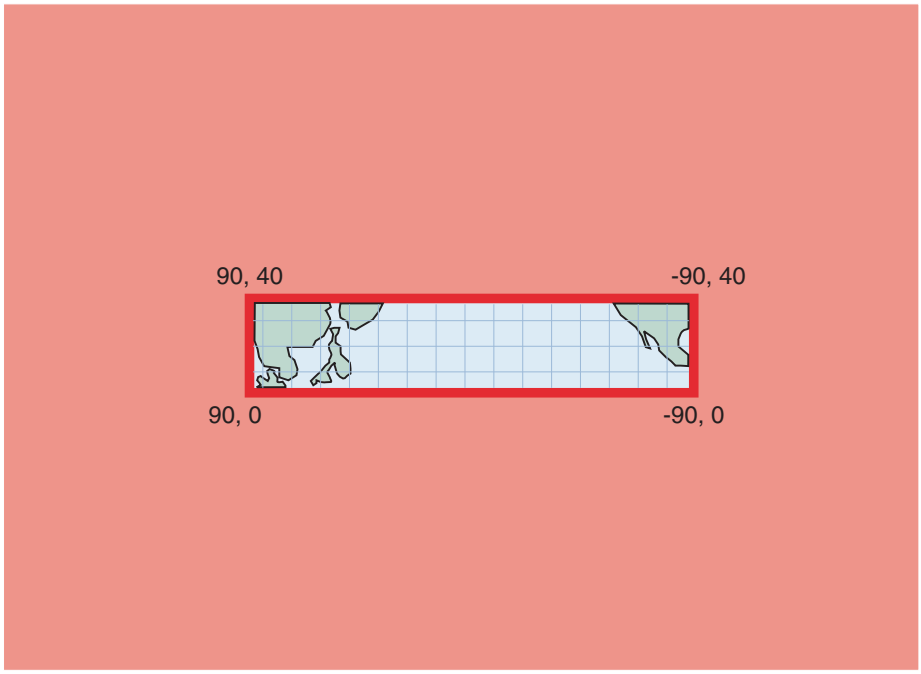


図 39. 180 度子午線をまたぐポリゴン—補完する領域

極点を囲むポリゴン

Spatial Extender の場合、平面地図投影で端の部分を扱うため、地図にゆがみが生じ、ポリゴン中の極地を表現するのに、余分なエッジや頂点を追加しなければならなくなっています。

175 ページの図 40 は、Spatial Extender または Geodetic Data Management Feature を使用して、南極を囲むポリゴンをどのように扱えるかを示したものです。

```
POLYGON((–180 –90, 180 –90, 180 –60, –180 –60, –180 –90))
```

球体地球表現 (Geodetic Data Management Feature) では、南極を囲むポリゴンを、–60 度の緯線をたどる円として表現できます。

```
POLYGON((0 –60, –1 –60, –2 –60, ..., –179 –60, 180 –60, 179 –60, ..., 1 –60, 0 –60))
```

データを再投影して、南極全体とそれを囲む領域を地図上に目に見えるように表示すれば、この円がどのようなものになるかよくわかるはずです。

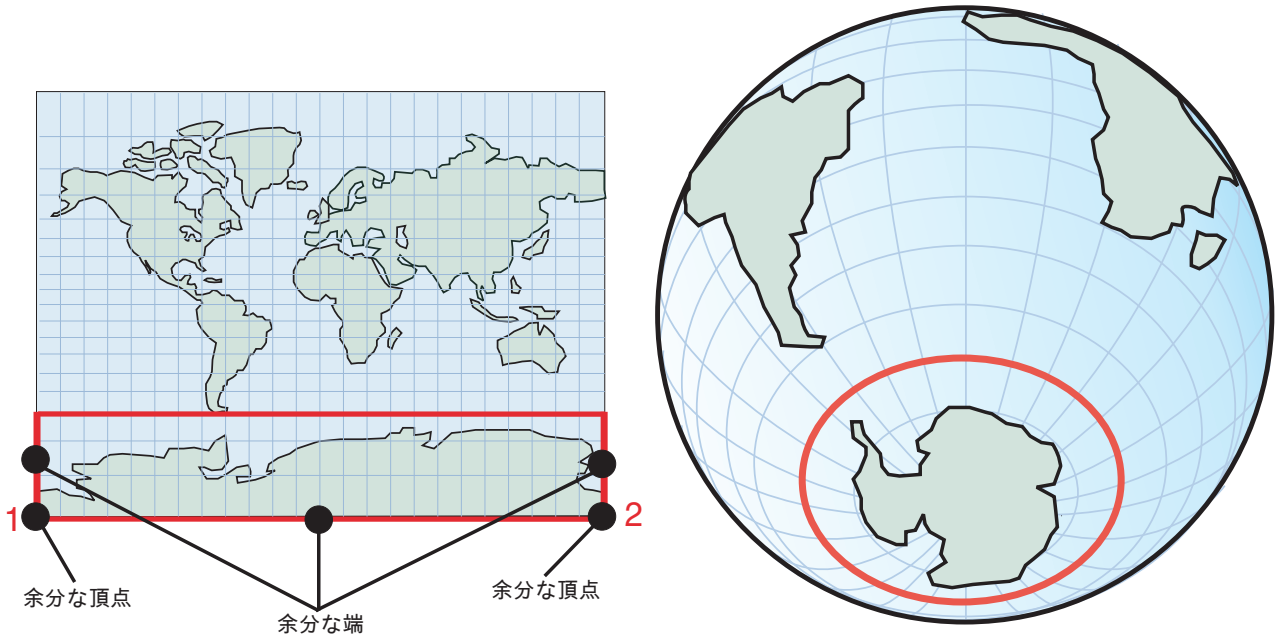


図 40. 極点を囲むポリゴン

上記の例の場合、適切な投影空間参照系を選択すれば、正しい結果を得ることができます。ただ、あらゆる状況に同時に対処できる投影法はありません。例えば、180 度子午線が端にならないよう投影をした場合でも、端は別の場所に移動するだけなので、それによって新たに問題となる領域ができることになります。

半球、赤道地帯、地球全体を表すポリゴン

球体地球表現では、地球表面上の広い領域にわたる距離と領域の計算で、より正確な結果を得ることができます。

半球のいずれか、赤道地帯、地球全体など、地球表面上の広い領域を表すポリゴンを使用する必要がある場合、Spatial Extender と Geodetic Data Management Feature では、処理方法が異なる点に注意してください。この場合、球体地球表現を利用すれば、距離や面積を正確に計算できますが、投影だと、どれほど注意深く行ったとしても、正確な計算はできません。

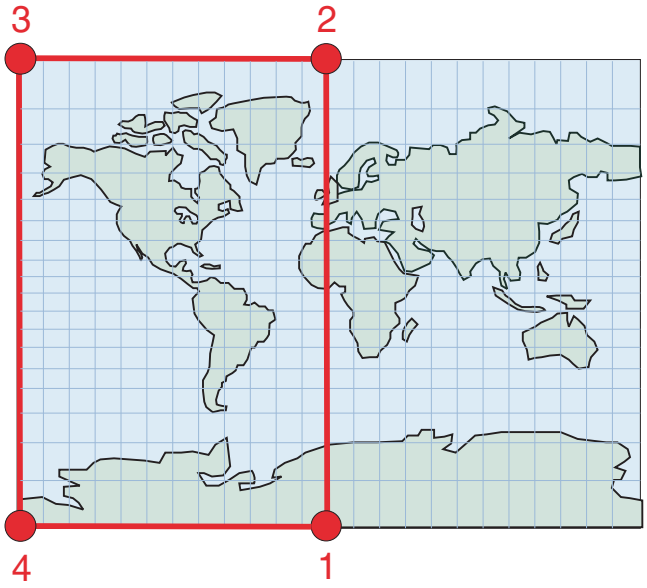
例えば、 176 ページの図 41 は、西半球を定義するポリゴンを、平面地球表現 (Spatial Extender) と球体地球表現 (Geodetic Data Management Feature) の両方で示したものです。

- 176 ページの図 41 の上に示した 平面地球表現では、西半球を表す 4 つの座標が事前割り当てテキスト・フォーマットで 'POLYGON((0 -90, 0 90, -180 90, 180 -90, 0 -90))' になります。
- 球体地球表現では、西半球を表す 4 つの座標が事前割り当てテキスト・フォーマットで 'POLYGON((0 0, 0 90, 180 0, 0 -90, 0 0))' になります。この 4 つの座標は、地球の第 0 子午線と、その正反対に位置する 180 度子午線に沿ったリングを定義します。

同じ 4 つのポイントを、反対の順序で指定すると、東半球が定義されます。

- 平面地球表現では、東半球は 'POLYGON((0 -90, 180 -90, 180 90, 0 90, 0 -90))' で表されます。
- 球体地球表現では、東半球は 'POLYGON((0 -90, 180 0, 0 90, 0 0, 0 -90))' で表されます。

西半球、平面地球表現
Polygon ((0 -90, 0 90, -180 90, 180 -90, 0 -90))



西半球、球体地球表現
Polygon ((0 0, 0 90, 180 0, 0 -90, 0 0))

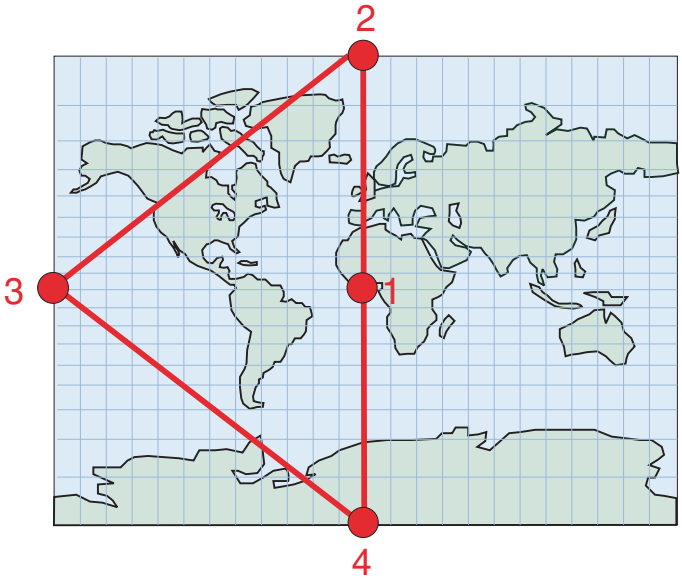


図 41. 半球を表すポリゴン

178 ページの図 42 は、赤道地帯を定義するポリゴンの座標を、平面地球表現 (Spatial Extender) と、球体地球表現 (Geodetic Data Management Feature) で示したものです。

- 178 ページの図 42 の上側は、赤道地帯を、平面地図表現により、事前割り当てテキスト・フォーマットの座標 '((180 –60, 180 60, –180 60, –180 –60, 180 –60))' -60))' で表したものです。
- 球体地球表現では、 178 ページの図 42 の下側に示したように、赤道地帯を、2 つのリングの排他領域を定義することによって表現します。

```
'MULTIPOLYGON(((0 60, -120 60, 120 60, 0 60)),
   ((0 -60, 120 -60, -120 -60, 0 -60)))'
```

  わかりやすくするため、各リングのポイントは 3 つだけにしてあります。実際には、60 度、あるいは -60 度の緯線をより正確にたどるためには、中間のポイントも追加する必要があります。 1 つ目のリング ((0 60, -120 60, 120 60, 0 60)) では、頂点を、60 度の緯線の南の領域を定義できる順序で指定します。 2 つ目のリング ((0 -60, 120 -60, -120 -60, 0 -60)) では、-60 度の緯線の北の領域を定義できる順序で頂点を指定します。

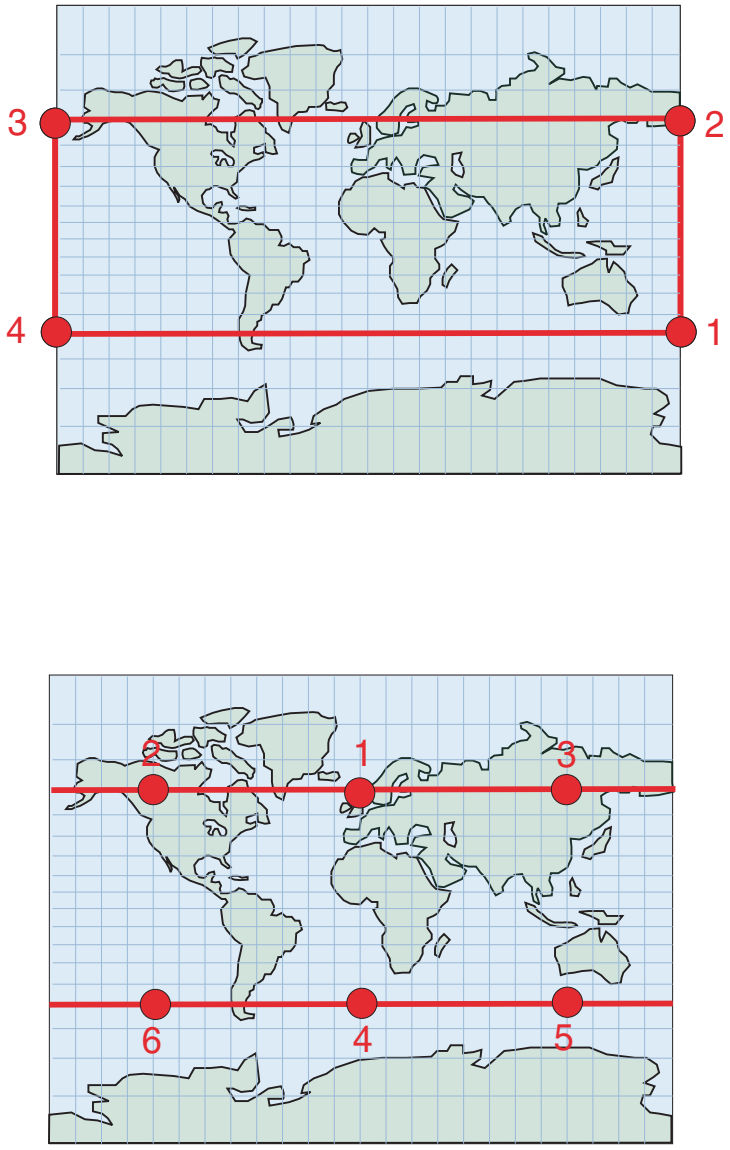


図 42. 赤道地帯を表すポリゴン

179 ページの図 43 は、地球全体を定義するポリゴンを、平面地球表現 (Spatial Extender) と 球体地球表現 (Geodetic Data Management Feature) で示したものです。いずれの表現でも、地球全体は、事前割り当てテキスト・フォーマットで 'POLYGON((-180 -90, 180 -90, 180 90, -180 90, -180 -90))' という同じポリゴンで定義されます。

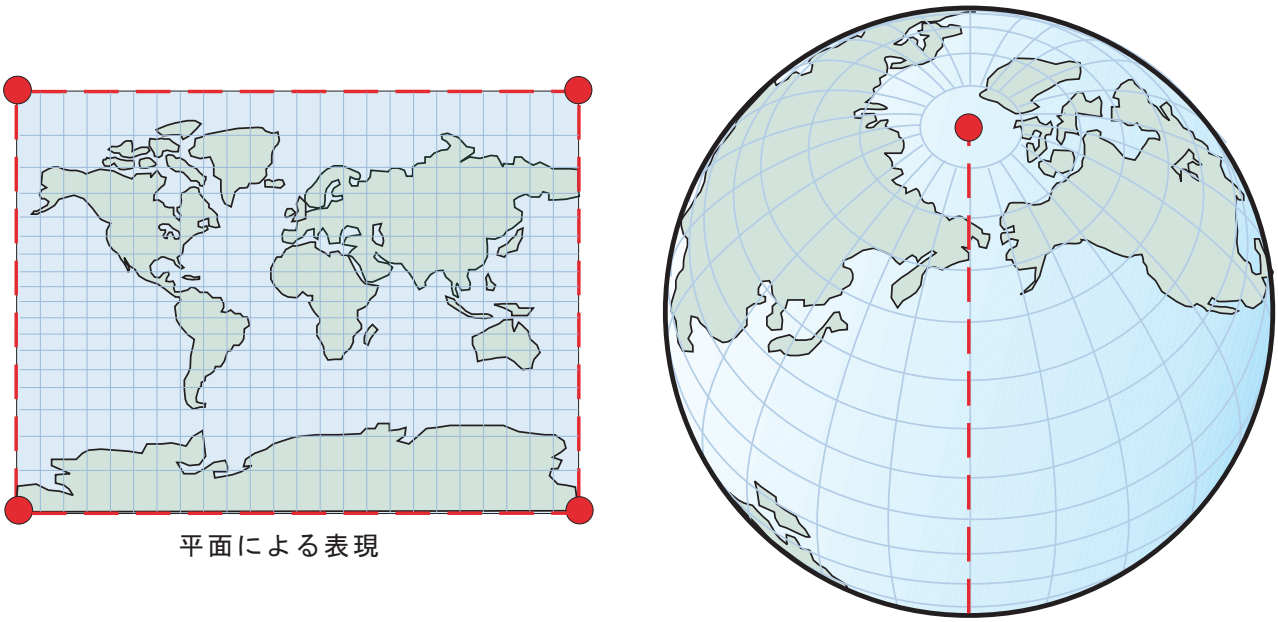


図 43. 地球全体を表すポリゴン

DB2 Geodetic Data Management Feature によってサポートされる空間処理関数

DB2 Spatial Extender は、ESRI によって提供されている関数ライブラリーを基盤にし、DB2 Geodetic Data Management Feature は、Hipparchus 関数ライブラリーを基盤にして作られています。 ESRI ライブラリーと Hipparchus の機能の違いにより、いくつかの関数の機能に若干の違いが生じます。以下の表は、Geodetic Data Management Feature がサポートしている Spatial Extender の関数をまとめたもので、Spatial Extender と Geodetic Data Management Feature でフィーチャーに違いがある場合は、それについても示しています。空間処理関数の使い方や構文の詳細については、個々の空間処理関数について説明している箇所を参照してください。

表 24. Geodetic Data Management Feature の関数サポート

関数	DB2 Geodetic Data Management Feature がサポートするかどうか	DB2 Geodetic Data Management Feature での動作の違い
EnvelopesIntersect	はい	なし
MBR 集約	いいえ	該当なし
ST_AppendPoint	いいえ	該当なし
ST_Area	はい	デフォルトの測定単位がメートル。
ST_AsBinary	はい	なし

表 24. *Geodetic Data Management Feature* の関数サポート (続き)

関数	DB2 Geodetic Data Management Feature がサポートするかどうか	DB2 Geodetic Data Management Feature での動作の違い
ST_AsGML	はい	なし
ST_AsShape	はい	なし
ST_AsText	はい	なし
ST_Boundary	いいえ	該当なし
ST_Buffer	はい	ポイントと複数ポイントに関してのみサポートされている。距離は負の値にできる。デフォルトの測定単位がメートル。
ST_Centroid	いいえ	該当なし
ST_ChangePoint	いいえ	該当なし
ST_Contains	はい	2 つの形状が同じ測地参照系 (SRS) で表現される必要がある。
ST_ConvexHull	いいえ	該当なし
ST_CoordDim	はい	なし
ST_Crosses	いいえ	該当なし
ST_Difference	はい	折れ線、複数折れ線に関してはサポートされていない。2 つの形状が同じ測地参照系 (SRS) で表現される必要がある。戻される形状のディメンションは、入力された形状と同じになる。
ST_Dimension	はい	なし
ST_Disjoint	はい	なし
ST_Distance	はい	*測地距離* を戻す。2 つの形状が同じ測地参照系 (SRS) で表現される必要がある。デフォルトの測定単位がメートル。
ST_Edge_GC_USA	はい	なし
ST_Endpoint	はい	なし
ST_Envelope	はい	エンベロープは、形状の最小の外接円 (MBC) を囲むポリゴンになる。
ST_EnvIntersects	はい	なし
ST_EqualCoordsys	はい	なし
ST_Equals	いいえ	該当なし
ST_EqualSRS	はい	なし
ST_ExteriorRing	はい	なし
ST_FindMeasure または ST_LocateAlong	いいえ	該当なし
ST_Generalize	はい	しきい値の単位がメートル。
ST_GeomCollection	いいえ	該当なし
ST_GeomCollFromTxt	いいえ	該当なし
ST_GeomCollFromWKB	いいえ	該当なし
ST_Geometry	はい	なし
ST_GeometryN	はい	なし

表 24. *Geodetic Data Management Feature* の関数サポート (続き)

関数	DB2 Geodetic Data Management Feature がサポートするかどうか	DB2 Geodetic Data Management Feature での動作の違い
ST_GeometryType	はい	なし
ST_GeomFromText	はい	なし
ST_GeomFromWKB	はい	なし
ST_GetIndexParms	いいえ	該当なし
ST_InteriorRingN	はい	なし
ST_Intersection	はい	戻される形状のディメンションは、2 つの折れ線の交点のディメンションが 0 の場合を除き、入力された形状のうちディメンションの低い方に合わされる。
ST_Intersects	はい	2 つの形状が同じ測地参照系 (SRS) で表現される必要がある。
ST_Is3d	はい	なし
ST_IsClosed	はい	なし
ST_IsEmpty	はい	なし
ST_IsMeasured	はい	なし
ST_IsRing	いいえ	該当なし
ST_IsSimple	いいえ	該当なし
ST_IsValid	はい	なし
ST_Length	はい	デフォルトの測定単位がメートル。
ST_LineFromText	はい	なし
ST_LineFromWKB	はい	なし
ST_LineString	はい	なし
ST_LineStringN	はい	なし
ST_M	はい	なし
ST_MaxM	はい	なし
ST_MaxX	はい	最小の外接円 (MBC) の最大の X 値を戻す。 **注:** MBC が日付変更線をまたぐ場合は、ST_MaxX 値は ST_MinX より小さくなる。MBC が北極点あるいは南極点を含む場合、ST_MinX は –180、ST_MaxX は 180 になる。
ST_MaxY	はい	MBC の最大の Y 値を戻す。 **注:** MBC が北極点を含む場合、ST_MaxY の値は 90 になる。
ST_MaxZ	はい	なし
ST_MBR	はい	MBR は、形状の MBC を囲む形状。
ST_MBRIntersects	はい	なし
ST_MeasureBetween または ST_LocateBetween	いいえ	該当なし
ST_MidPoint	はい	なし
ST_MinM	はい	なし

表 24. *Geodetic Data Management Feature* の関数サポート (続き)

関数	DB2 Geodetic Data Management Feature がサポートするかどうか	DB2 Geodetic Data Management Feature での動作の違い
ST_MinX	はい	MBC の最小の X 値を戻す。 **注:** MBC が日付変更線をまたぐ場合は、ST_MinX 値は ST_MaxX 値より大きくなる。MBC が北極点あるいは南極点を含む場合、ST_MinX は –180、ST_MaxX は 180 になる。
ST_MinY	はい	MBC の最小の Y 値を戻す。 **注:** MBC が南極点を含む場合、ST_MinY の値は –90 になる。
ST_MinZ	はい	なし
ST_MLineFromText	はい	なし
ST_MLineFromWKB	はい	なし
ST_MPointFromText	はい	なし
ST_MPointFromWKB	はい	なし
ST_MPolyFromText	はい	なし
ST_MPolyFromWKB	はい	なし
ST_MultiLineString	はい	なし
ST_MultiPoint	はい	なし
ST_MultiPolygon	はい	なし
ST_NumGeometries	はい	なし
ST_NumInteriorRing	はい	なし
ST_NumLineStrings	はい	なし
ST_NumPoints	はい	なし
ST_NumPolygons	はい	なし
ST_Overlaps	いいえ	該当なし
ST_Perimeter	はい	デフォルトの測定単位がメートル。
ST_PerpPoints	いいえ	該当なし
ST_Point	はい	なし
ST_PointFromText	はい	なし
ST_PointFromWKB	はい	なし
ST_PointN	はい	なし
ST_PolyFromText	はい	なし
ST_PolyFromWKB	はい	なし
ST_PointOnSurface	はい	なし
ST_Polygon	はい	なし
ST_PolygonN	はい	なし
ST_Relate	いいえ	該当なし
ST_RemovePoint	いいえ	該当なし
ST_SrsId または ST_SRID	はい	なし

表 24. *Geodetic Data Management Feature* の関数サポート (続き)

関数	DB2 Geodetic Data Management Feature がサポートするかどうか	DB2 Geodetic Data Management Feature での動作の違い
ST_SrsName	はい	なし
ST_StartPoint	はい	なし
ST_SymDifference	はい	折れ線、複数折れ線に関してはサポートされていない。戻される形状のディメンションは、入力された形状と同じになる。2 つの形状が同じ測地参照系 (SRS) で表現される必要がある。
ST_ToGeomColl	いいえ	該当なし
ST_ToLineString	はい	なし
ST_ToMultiLine	はい	なし
ST_ToMultiPoint	はい	なし
ST_ToPoint	はい	なし
ST_ToPolygon	はい	なし
ST_Touches	いいえ	該当なし
ST_Transform	はい	なし。**注 :** 座標変換はポイントごとに行われる。地理座標システムと非投影平面座標システムの間でトランスフォームする場合は、ポリゴンや折れ線の中に、180 度子午線をまたぐものや、北極点か南極点 (あるいはその両方) を囲むものがないか慎重に確認する。 Spatial Extender と Geodetic Data Management Feature では、そのようなケースへの対処が異なるため、平面地球座標システムでは有効であった形状が、球体地球では有効でなくなる可能性がある (逆もまた同様)。詳しくは、 170 ページの『平面地球を使用した場合と球体地球を使用した場合の違い』 を参照してください。
ST_Union	はい	2 つの形状が同じ測地参照系 (SRS) で表現される必要がある。
ST_Within	はい	2 つの形状が同じ測地参照系 (SRS) で表現される必要がある。
ST_WKBToSQL	はい	なし
ST_WKTToSQL	はい	なし
ST_X	はい	なし
ST_Y	はい	なし
ST_Z	はい	なし
和集約	いいえ	該当なし

DB2 Geodetic Data Management Feature のストアード・プロシージャーとカタログ・ビュー

DB2 Geodetic Data Management Feature は、DB2 Spatial Extender と同じカタログ・ビューをサポートし、空間ストアード・プロシージャーのサブセットをサポートします。

ただし、Geodetic Data Management Feature は以下のストアード・プロシージャーはサポートしていません。

- ST_disable_autogeocoding
- ST_enable_autogeocoding
- ST_register_geocoder
- ST_remove_geocoding_setup
- ST_run_geocoding
- ST_setup_geocoding
- ST_unregister_geocoder

Geodetic Data Management Feature は、DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビューに表示される 318 の定義済み測地参照系を提供しています。完全なリストを参照してください。

DB2 Geodetic Data Management Feature によってサポートされる測地系

説明したとおり、*測地系*は回転楕円面の、地球の中心に対する相対的な位置を定義する値のセットです。空間参照系 (SRS) は、測地系と回転楕円面を関連付けるパラメーターのセットで、空間参照系 ID (SRID) によって識別されます。 185 ページの表 26 は、DB2 Geodetic Data Management Feature が提供する定義済み測地系のリストです。 オフセット値とスケール係数は、すべての定義済み測地 SRS について同じです。下の表に値を示しておきます。

表 25. 定義済み測地 SRS のオフセット値、あるいはスケール係数の値

SRS パラメーター	**値**
xOffset	–180
yOffset	–90
zOffset	–50000
mOffset	–1000
xScale	5965232
yScale	5965232
zScale	1000
mScale	1000

yScale は常に xScale と同じです。

ユーザーは、使用する空間参照系について、表 26 にリストしたいずれの測地系も選択できます。ただ、できればデータに最も適合したものを選択するようにすべきです。例えば、最も一般的に使用されている測地系の 1 つである World Geodetic System 1984 (WGS 1984) では、地球の中心を原点とし、地球上の各地点の座標を決めています。WGS 1984 は、地球中心の測地系の一種です。それに対し、North American 1927 測地系のような特定の地域に対応する測地系は、地上の一点を原点として地域（この場合は北米) の各地点の座標を決めています。この種の測地系は、モデル化する地域については正確に表せるのですが、地球上の様々な地点について操作を行う場合には、どうしても地球中心の測地系が必要になります。

表 26. 測地系、回転楕円面と SRID

SRID	測地系の名前	参照回転楕円面
2000000000	WGS 1984	WGS 1984
2000000001	Abidjan 1987	Clarke 1880 (RGS)
2000000002	Accra	War Office
2000000003	Adindan	Clarke 1880 (RGS)
2000000004	Afgooye	Krasovsky 1940
2000000005	Agadez	Clarke 1880 (IGN)
2000000006	Australian Geodetic Datum 1966	Australian
2000000007	Australian Geodetic Datum 1984	Australian
2000000008	Ain el Abd 1970	International 1924
2000000009	Airy 1830	Airy 1830
2000000010	Airy Modified	Airy Modified
2000000011	Alaskan Islands	Clarke 1866
2000000012	Amersfoort	Bessel 1841
2000000013	Anguilla 1957	Clarke 1880 (RGS)
2000000014	Anna 1 Astro 1965	Australian
2000000015	Antigua Astro 1943	Clarke 1880 (RGS)
2000000016	Aratu	International 1924
2000000017	Arc 1950	Clarke 1880 (Arc)
2000000018	Arc 1960	Clarke 1880 (RGS)
2000000019	Ascension Island 1958	International 1924
2000000020	Assumed Geographic (形状ファイルは NAD27、PRJ なし)	Clarke 1866
2000000021	Astronomical Station 1952	International 1924
2000000022	ATF (Paris)	Plessis 1817
2000000023	Average Terrestrial System 1977	ATS 1977
2000000024	Australian National	Australian
2000000025	Ayabelle Lighthouse	Clarke 1880 (RGS)
2000000026	Bab South Astro (Bablethuap Is, Republic of Palau)	Clarke 1866
2000000027	Barbados 1938	Clarke 1880 (RGS)
2000000028	Batavia	Bessel 1841
2000000029	Batavia (Jakarta)	Bessel 1841
2000000030	Astro Beacon E 1945	International 1924
2000000031	Beduaram	Clarke 1880 (IGN)
2000000032	Beijing 1954	Krasovsky 1940
2000000033	Reseau National Belge 1950	International 1924
2000000034	Belge 1950 (Brussels)	International 1924
2000000035	Reseau National Belge 1972	International 1924
2000000036	Bellevue (IGN)	International 1924
2000000037	Bermuda 1957	Clarke 1866
2000000038	Bern 1898	Bessel 1841
2000000039	Bern 1898 (Bern)	Bessel 1841

表 26. 測地系、回転楕円面と SRID (続き)

SRID	測地系の名前	参照回転楕円面
2000000040	Bern 1938	Bessel 1841
2000000041	Bessel 1841	Bessel 1841
2000000042	Bessel Modified	Bessel Modified
2000000043	Bessel Namibia	Bessel Namibia
2000000044	Bissau	International 1924
2000000045	Bogota	International 1924
2000000046	Bogota (Bogota)	International 1924
2000000047	Bukit Rimpah	Bessel 1841
2000000048	Camacupa	Clarke 1880 (RGS)
2000000049	Campo Inchauspe	International 1924
2000000050	Camp Area Astro	International 1924
2000000051	Canton Astro 1966	International 1924
2000000052	Cape	Clarke 1880 (Arc)
2000000053	Cape Canaveral	Clarke 1866
2000000054	Carthage	Clarke 1880 (IGN)
2000000055	Carthage (角度)	Clarke 1880 (IGN)
2000000056	Carthage (Paris)	Clarke 1880 (IGN)
2000000057	CH 1903	Bessel 1841
2000000058	CH 1903+	Bessel 1841
2000000059	Chatham Island Astro 1971	International 1924
2000000060	Chos Malal 1914	International 1924
2000000061	Swiss Terrestrial Ref.Frame 1995	GRS 1980
2000000062	Chua	International 1924
2000000063	Clarke 1858	Clarke 1858
2000000064	Clarke 1866	Clarke 1866
2000000065	Clarke 1866 (Michigan)	Clarke 1866 (Michigan)
2000000066	Clarke 1880	Clarke 1880
2000000067	Clarke 1880 (Arc)	Clarke 1880 (Arc)
2000000068	Clarke 1880 (Benoit)	Clarke 1880 (Benoit)
2000000069	Clarke 1880 (IGN)	Clarke 1880 (IGN)
2000000070	Clarke 1880 (RGS)	Clarke 1880 (RGS)
2000000071	Clarke 1880 (SGA)	Clarke 1880 (SGA)
2000000072	Conakry 1905	Clarke 1880 (IGN)
2000000073	Corrego Alegre	International 1924
2000000074	Cote d'Ivoire	Clarke 1880 (IGN)
2000000075	Dabola 1981	Clarke 1880 (RGS)
2000000076	Datum 73	International 1924
2000000077	Dealul Piscului 1933 (Romania)	International 1924
2000000078	Dealul Piscului 1970 (Romania)	Krasovsky 1940
2000000079	Deception Island	Clarke 1880 (RGS)
2000000080	Deir ez Zor	Clarke 1880 (IGN)
2000000081	Deutsche Hauptdreiecksnetz	Bessel 1841
2000000082	Dominica 1945	Clarke 1880 (RGS)
2000000083	DOS 1968	International 1924
2000000084	Astro DOS 71/4	International 1924
2000000085	Douala	Clarke 1880 (IGN)
2000000086	Easter Island 1967	International 1924
2000000087	European Datum 1950	International 1924
2000000088	European Datum 1950 (ED77)	International 1924
2000000089	European Datum 1987	International 1924
2000000090	Egypt 1907	Helmert 1906
2000000091	Estonia 1937	Bessel 1841
2000000092	Estonia 1992	GRS 1980

表 26. 測地系、回転楕円面と SRID (続き)

SRID	測地系の名前	参照回転楕円面
2000000093	European Terrestrial Ref.Frame 1989	WGS 1984
2000000094	European 1979	International 1924
2000000095	European Libyan Datum 1979	International 1924
2000000096	Everest 1830	Everest 1830
2000000097	Everest (Bangladesh)	Everest Adjustment 1937
2000000098	Everest (Definition 1962)	Everest (Definition 1962)
2000000099	Everest (Definition 1967)	Everest (Definition 1967)
2000000100	Everest (Definition 1975)	Everest (Definition 1975)
2000000101	Everest (India and Nepal)	Everest (Definition 1962)
2000000102	Everest 1830 Modified	Everest 1830 Modified
2000000103	Everest Modified 1969	Everest Modified 1969
2000000104	Fahud	Clarke 1880 (RGS)
2000000105	Final Datum 1958	Clarke 1880 (RGS)
2000000106	Fischer 1960	Fischer 1960
2000000107	Fischer 1968	Fischer 1968
2000000108	Fischer Modified	Fischer Modified
2000000109	Fort Thomas 1955	Clarke 1880 (RGS)
2000000110	Gandajika 1970	International 1924
2000000111	Gan 1970	International 1924
2000000112	Garoua	Clarke 1880 (IGN)
2000000113	Geocentric Datum of Australia 1994	GRS 1980
2000000114	GEM 10C Gravity Potential Model	GEM 10C
2000000115	Greek Geodetic Ref.System 1987	GRS 1980
2000000116	Graciosa Base SW 1948	International 1924
2000000117	Greek	Bessel 1841
2000000118	Greek (Athens)	Bessel 1841
2000000119	Grenada 1953	Clarke 1880 (RGS)
2000000120	GRS 1967	GRS 1967
2000000121	GRS 1980	GRS 1980
2000000122	Guam 1963	Clarke 1866
2000000123	Gunung Segara	Bessel 1841
2000000124	GUX 1 Astro	International 1924
2000000125	Guyane Francaise	International 1924
2000000126	Hanoi 1972	Krasovsky 1940
2000000127	Hartebeesthoek 1994	WGS 1984
2000000128	Helmert 1906	Helmert 1906
2000000129	Herat North	International 1924
2000000130	Hermannskogel	Bessel 1841
2000000131	Hito XVIII 1963	International 1924
2000000132	Hjorsey 1955	International 1924
2000000133	Hong Kong 1963	International 1924
2000000134	Hong Kong 1980	International 1924
2000000135	Hough 1960	Hough 1960
2000000136	Hungarian Datum 1972	GRS 1967
2000000137	Hu Tzu Shan	International 1924
2000000138	Indian 1954	Everest Adjustment 1937

表 26. 測地系、回転楕円面と SRID (続き)

SRID	測地系の名前	参照回転楕円面
2000000139	Indian 1960	Everest Adjustment 1937
2000000140	Indian 1975	Everest Adjustment 1937
2000000141	Indonesian National	Indonesian National
2000000142	Indonesian Datum 1974	Indonesian
2000000143	International 1927	International 1924
2000000144	International 1967	International 1967
2000000145	IRENET95	GRS 1980
2000000146	Israel	GRS 1980
2000000147	ISTS 061 Astro 1968	International 1924
2000000148	ISTS 073 Astro 1969	International 1924
2000000149	Jamaica 1875	Clarke 1880
2000000150	Jamaica 1969	Clarke 1866
2000000151	Japan Geodetic Datum 2000	GRS 1980
2000000152	Johnston Island 1961	International 1924
2000000153	Kalianpur 1880	Everest 1830
2000000154	Kalianpur 1937	Everest Adjustment 1937
2000000155	Kalianpur 1962	Everest (Definition 1962)
2000000156	Kalianpur 1975	Everest (Definition 1975)
2000000157	Kandawala	Everest Adjustment 1937
2000000158	Kerguelen Island 1949	International 1924
2000000159	Kertau	Everest 1830 Modified
2000000160	Kartastokoordinaattijarjestelma	International 1924
2000000161	Kuwait Oil Company	Clarke 1880 (RGS)
2000000162	Korean Datum 1985	Bessel 1841
2000000163	Korean Datum 1995	WGS 1984
2000000164	Krasovsky 1940	Krasovsky 1940
2000000165	Kuwait Utility	GRS 1980
2000000166	Kusaie Astro 1951	International 1924
2000000167	Lake	International 1924
2000000168	La Canoa	International 1924
2000000169	L.C. 5 Astro 1961	Clarke 1866
2000000170	Leigon	Clarke 1880 (RGS)
2000000171	Liberia 1964	Clarke 1880 (RGS)
2000000172	Datum Lisboa Bessel	Bessel 1841
2000000173	Datum Lisboa Hayford	International 1924
2000000174	Lisbon	International 1924
2000000175	Lisbon (Lisbon)	International 1924
2000000176	LKS 1994	GRS 1980
2000000177	Locodjo 1965	Clarke 1880 (RGS)
2000000178	Loma Quintana	International 1924
2000000179	Lome	Clarke 1880 (IGN)
2000000180	Luzon 1911	Clarke 1866
2000000181	Madrid 1870 (Madrid Prime Merid.)	Struve 1860
2000000182	Madzansua	Clarke 1866
2000000183	Mahe 1971	Clarke 1880 (RGS)
2000000184	Majuro (Republic of Marshall Is.)	Clarke 1866

表 26. 測地系、回転楕円面と SRID (続き)

SRID	測地系の名前	参照回転楕円面
2000000185	Makassar	Bessel 1841
2000000186	Makassar (Jakarta)	Bessel 1841
2000000187	Malongo 1987	International 1924
2000000188	Manoca	Clarke 1880 (RGS)
2000000189	Massawa	Bessel 1841
2000000190	Merchich	Clarke 1880 (IGN)
2000000191	Merchich (角度)	Clarke 1880 (IGN)
2000000192	Militar-Geographische Institut	Bessel 1841
2000000193	MGI (Ferro)	Bessel 1841
2000000194	Mhast	International 1924
2000000195	Midway Astro 1961	International 1924
2000000196	Minna	Clarke 1880 (RGS)
2000000197	Monte Mario	International 1924
2000000198	Monte Mario (Rome)	International 1924
2000000199	Montserrat Astro 1958	Clarke 1880 (RGS)
2000000200	Mount Dillon	Clarke 1858
2000000201	Moznet	WGS 1984
2000000202	M'poraloko	Clarke 1880 (IGN)
2000000203	North American Datum 1927	Clarke 1866
2000000204	NAD 1927 CGQ77	Clarke 1866
2000000205	NAD 1927 (1976)	Clarke 1866
2000000206	North American Datum 1983	GRS 1980
2000000207	NAD 1983 (Canadian Spatial Ref. System)	GRS 1980
2000000208	North American Datum 1983 (HARN)	GRS 1980
2000000209	NAD Michigan	Clarke 1866 (Michigan)
2000000210	Nahrwan 1967	Clarke 1880 (RGS)
2000000211	Naparima 1955	International 1924
2000000212	Naparima 1972	International 1924
2000000213	Nord de Guerre (Paris)	Plessis 1817
2000000214	National Geodetic Network (Kuwait)	WGS 1984
2000000215	NGO 1948	Bessel Modified
2000000216	NGO 1948 (Oslo)	Bessel Modified
2000000217	Nord Sahara 1959	Clarke 1880 (RGS)
2000000218	NSWC 9Z-2	NWL 9D
2000000219	Nouvelle Triangulation Francaise (degrees)	Clarke 1880 (IGN)
2000000220	NTF (Paris) (グラジアン)	Clarke 1880 (IGN)
2000000221	NWL 9D Transit Precise Ephemeris	NWL 9D
2000000222	New Zealand Geodetic Datum 1949	International 1924
2000000223	New Zealand Geodetic Datum 2000	GRS 1980
2000000224	Observatario	Clarke 1866
2000000225	Observ. Meteorologico 1939	International 1924
2000000226	Old Hawaiian	Clarke 1866
2000000227	Oman	Clarke 1880 (RGS)
2000000228	OSGB 1936	Airy 1830
2000000229	OSGB 1970 (SN)	Airy 1830
2000000230	OSU 1986 Geoidal Model	OSU 86F
2000000231	OSU 1991 Geoidal Model	OSU 91A
2000000232	OS (SN) 1980	Airy 1830
2000000233	Padang 1884	Bessel 1841
2000000234	Padang 1884 (Jakarta)	Bessel 1841
2000000235	Palestine 1923	Clarke 1880 (Benoit)
2000000236	Pampa del Castillo	International 1924
2000000237	PDO Survey Datum 1993	Clarke 1880 (RGS)

表 26. 測地系、回転楕円面と SRID (続き)

SRID	測地系の名前	参照回転楕円面
2000000238	Pico de Las Nieves	International 1924
2000000239	Pitcairn Astro 1967	International 1924
2000000240	Plessis 1817	Plessis 1817
2000000241	Pohnpei (Fed. States of Micronesia)	Clarke 1866
2000000242	Point 58	Clarke 1880 (RGS)
2000000243	Pointe Noire	Clarke 1880 (IGN)
2000000244	Porto Santo 1936	International 1924
2000000245	POSGAR	GRS 1980
2000000246	Provisional South Amer. Datum 1956	International 1924
2000000247	Puerto Rico	Clarke 1866
2000000248	Pulkovo 1942	Krasovsky 1940
2000000249	Pulkovo 1995	Krasovsky 1940
2000000250	Qatar 1974	International 1924
2000000251	Qatar 1948	Helmert 1906
2000000252	Qornoq	International 1924
2000000253	Rassadiran	International 1924
2000000254	REGVEN	GRS 1980
2000000255	Reunion	International 1924
2000000256	Reseau Geodesique Francais 1993	GRS 1980
2000000257	RT38	Bessel 1841
2000000258	RT38 (Stockholm)	Bessel 1841
2000000259	RT 1990	Bessel 1841
2000000260	S-42 Hungary	Krasovsky 1940
2000000261	South American Datum 1969	GRS 1967 Truncated
2000000262	Samboja	Bessel 1841
2000000263	American Samoa 1962	Clarke 1866
2000000264	Santo DOS 1965	International 1924
2000000265	Sao Braz	International 1924
2000000266	Sapper Hill 1943	International 1924
2000000267	Schwarzeck	Bessel Namibia
2000000268	Segora	Bessel 1841
2000000269	Selvagem Grande 1938	International 1924
2000000270	Serindung	Bessel 1841
2000000271	Sierra Leone 1924	War Office
2000000272	Sierra Leone 1960	Clarke 1880 (RGS)
2000000273	Sierra Leone 1968	Clarke 1880 (RGS)
2000000274	SIRGAS	GRS 1980
2000000275	South Yemen	Krasovsky 1940
2000000276	Authalic sphere	Sphere
2000000277	Authalic sphere (ARC/INFO)	Sphere ARC INFO
2000000278	Struve 1860	Struve 1860
2000000279	St. George Island (Alaska)	Clarke 1866
2000000280	St. Kitts 1955	Clarke 1880 (RGS)
2000000281	St. Lawrence Island (Alaska)	Clarke 1866
2000000282	St. Lucia 1955	Clarke 1880 (RGS)
2000000283	St. Paul Island (Alaska)	Clarke 1866
2000000284	St. Vincent 1945	Clarke 1880 (RGS)
2000000285	Sudan	Clarke 1880 (IGN)
2000000286	South Asia Singapore	Fischer Modified
2000000287	S-JTSK	Bessel 1841
2000000288	S-JTSK (Ferro)	Bessel 1841
2000000289	Tananarive 1925	International 1924
2000000290	Tananarive 1925 (Paris)	International 1924

表 26. 測地系、回転楕円面と SRID (続き)

SRID	測地系の名前	参照回転楕円面
2000000291	Tern Island Astro 1961	International 1924
2000000292	Tete	Clarke 1866
2000000293	Timbalai 1948	Everest (Definition 1967)
2000000294	TM65	Airy Modified
2000000295	TM75	Airy Modified
2000000296	Tokyo	Bessel 1841
2000000297	Trinidad 1903	Clarke 1858
2000000298	Tristan Astro 1968	International 1924
2000000299	Trucial Coast 1948	Helmert 1906
2000000300	Viti Levu 1916	Clarke 1880 (RGS)
2000000301	Voirol 1875	Clarke 1880 (IGN)
2000000302	Voirol 1875 (角度)	Clarke 1880 (IGN)
2000000303	Voirol 1875 (Paris)	Clarke 1880 (IGN)
2000000304	Voirol Unifie 1960	Clarke 1880 (RGS)
2000000305	Voirol Unifie 1960 (角度)	Clarke 1880 (RGS)
2000000306	Voirol Unifie 1960 (Paris)	Clarke 1880 (RGS)
2000000307	Wake-Eniwetok 1960	Hough 1960
2000000308	Wake Island Astro 1952	International 1924
2000000309	Walbeck	Walbeck
2000000310	War Office	War Office
2000000311	WGS 1966	WGS 1966
2000000312	WGS 1972	WGS 1972
2000000313	WGS 1972 Transit Broadcast Ephemeris	WGS 1972
2000000314	Yacare	International 1924
2000000315	Yemen Nat'l Geodetic Network 1996	WGS 1984
2000000316	Yoff	Clarke 1880 (IGN)
2000000317	Zanderij	International 1924

測地回転楕円体

回転楕円体 (回転楕円面とも呼ばれる) は、特定地点での地球表面の形状を定義する地理座標システムの構成要素です。

DATUM で座標システムを定義する際には、以下の例に示すように、SPHEROID による回転楕円体 (面) の定義も行います。

```
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199432955]]
```

この情報を検索するには、DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューを使用できます。DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューの **DEFINITION** 列 には、表の **名前**、**半長軸**、**逆平坦化**といった列の値が含まれています。

第 20 章 ストアード・プロシージャー

このセクションでは、DB2 Spatial Extender をセットアップして、空間データを使用するプロジェクトを作成するために使用できる、DB2 Spatial Extender ストアード・プロシージャーに関する参照情報を提供しています。

DB2 Spatial Extender をセットアップするとき、または DB2 コントロール・センターや DB2 コマンド行プロセッサーからプロジェクトを作成するときに、これらのストアード・プロシージャーが暗黙的に呼び出されます。例えば、DB2 コントロール・センターの DB2 Spatial Extender ウィンドウから「**OK**」をクリックすると、DB2 はそのウィンドウに関連付けられている DB2 Spatial Extender ストアード・プロシージャーを呼び出します。

あるいは、アプリケーション・プログラム内で明示的に DB2 Spatial Extender ストアード・プロシージャーを呼び出すことができます。

ほとんどの DB2 Spatial Extender ストアード・プロシージャーでは、データベース上でそれらを呼び出す前に次のタスクを実行してください。

1. ページ・サイズが 8 KB で、最小で 500 ページのサイズがある SYSTEM TEMPORARY 表スペースがあることを確認します。そのような表スペースがない場合、作成方法の詳細について「*データベース: 管理の概念および構成リファレンス*」の『TEMPORARY 表スペースの作成』を参照してください。 これは、ST_enable_db ストアード・プロシージャーまたは db2se enable_db コマンドを正常に実行するための要件です。
2. ST_enable_db ストアード・プロシージャーを直接、または DB2 コントロール・センターを使用して呼び出すことによって、データベースで空間操作を行えるようにします。詳しくは、 216 ページの『ST_enable_db』を参照してください。

データベースに空間操作を行えるようにした後は、任意の DB2 Spatial Extender ストアード・プロシージャーをそのデータベースに対して (接続されている場合) 暗黙的または明示的に呼び出すことができます。

この章では、以下のすべての DB2 Spatial Extender ストアード・プロシージャーを説明しています。

© Copyright IBM Corp. 1998, 2009

- 218 ページの『ST_export_shape』
- 222 ページの『ST_import_shape』
- 230 ページの『ST_register_geocoder』
- 234 ページの『ST_register_spatial_column』
- 236 ページの『ST_remove_geocoding_setup』
- 238 ページの『ST_run_geocoding』
- 241 ページの『ST_setup_geocoding』
- 245 ページの『ST_unregister_geocoder』
- 246 ページの『ST_unregister_spatial_column』

ストアード・プロシージャーのインプリメンテーションは、DB2 Spatial Extender サーバーの db2gse ライブラリーにアーカイブされています。

ST_alter_coordsys

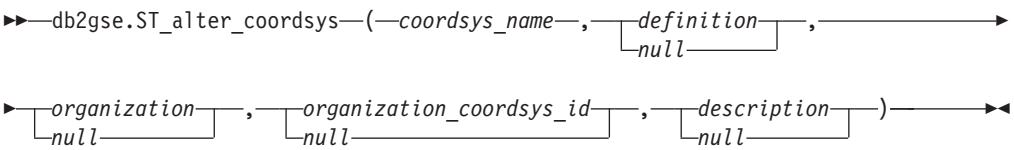
このストアード・プロシージャーは、データベース内の座標システム定義を更新するために使用します。このストアード・プロシージャーが処理されると、座標システムについての情報が DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューで更新されます。

重要: このストアード・プロシージャーの使用には注意が必要です。このストアード・プロシージャーを使用して座標システムの定義を変更した場合、この座標システムに基づく空間参照系に関連付けられた、既存の空間データがある場合、この空間データを気付かずに変更してしまう可能性があります。影響を受ける空間データがある場合、変更された空間データが依然として正確かつ有効であることを、確認する必要があります。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、DBADM 権限を持つ必要があります。

構文



パラメーターの説明

coordsys_name
座標システムを一意的に指定します。このパラメーターには NULL でない値を指定する必要があります。

coordsys_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

definition
座標システムを定義します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、座標システムの定義は変更されません。

このパラメーターのデータ・タイプは VARCHAR(2048) です。

organization
座標系を定義し、その定義を提供した団体の名前。たとえば、「European Petroleum Survey Group (EPSG)」など。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。

このパラメーターが NULL の場合、座標システムの団体は変更されません。このパラメーターが NULL でない場合、*organization_coordsys_id* パラメーターは NULL にできません。この場合、*organization* と *organization_coordsys_id* パラメーターを組み合わせて座標システムを一意的に識別します。

このパラメーターのデータ・タイプは VARCHAR(128) です。

organization_coordsys_id
organization パラメーターにリストされたエンティティーにより、この座標システムに割り当てられた数値 ID を指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。

このパラメーターが NULL の場合、*organization* パラメーターも NULL にする必要があります。この場合、その団体の座標システム ID は変更されません。このパラメーターが NULL でない場合、*organization* パラメーターは NULL にできません。この場合、*organization* と *organization_coordsys_id* パラメーターを組み合わせて座標システムを一意的に識別します。

このパラメーターのデータ・タイプは INTEGER です。

description
アプリケーションを説明することにより、座標システムを記述します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、座標システムに関する記述は変更されません。

このパラメーターのデータ・タイプは VARCHAR(256) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際

のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_alter_coordsys ストアード・プロシージャーを呼び出す方法を示しています。 この例は、DB2 CALL コマンドを使用して、NORTH_AMERICAN_TEST という名前の座標システムを変更します。 この CALL コマンドは、*coordsys_id* パラメーターに値 1002 を割り当てます。

```
call db2gse.ST_alter_coordsys('NORTH_AMERICAN_TEST',NULL,NULL,1002,NULL,?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_alter_srs

このストアード・プロシージャーは、データベース内の空間参照系の定義を更新するために使用します。このストアード・プロシージャーが処理されると、空間参照系についての情報が DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビューで更新されます。

内部的には、DB2 Spatial Extender は座標値を正の整数として保管します。 したがって計算中の、丸めエラーによる影響 (浮動小数点演算の実際の値によって大きく変わる) を減らすことができます。また、空間操作のパフォーマンスも大幅に改善できます。

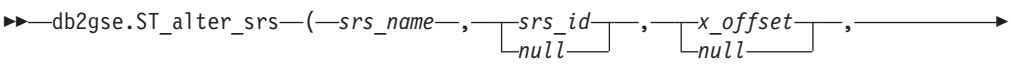
制約事項: ある空間参照系を使用する空間列が登録されている場合は、その空間参照系を変更することはできません。

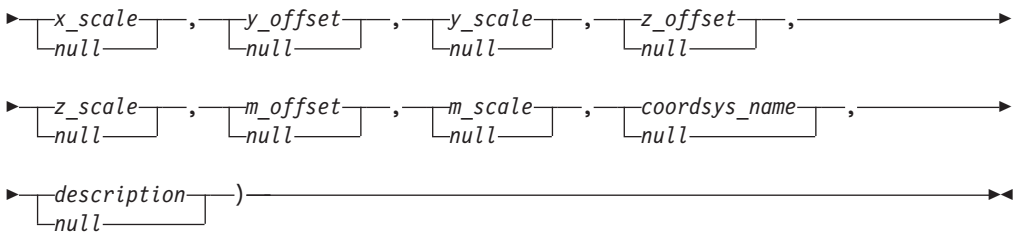
重要: このストアード・プロシージャーの使用には注意が必要です。このストアード・プロシージャーを使用して、空間参照系のオフセット、スケール、または *coordsys_name* パラメーターを変更すると、この空間参照系に関連付けられた既存の空間データがある場合、この空間データを気付かずに変更してしまう可能性があります。影響を受ける空間データがある場合、変更された空間データが依然として正確かつ有効であることを、確認する必要があります。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、DBADM 権限を持つ必要があります。

構文





パラメーターの説明

srs_name

空間参照系を示します。このパラメーターには NULL でない値を指定する必要があります。

srs_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

srs_id

空間参照系を一意的に識別します。この ID は、各種の空間処理関数の入力パラメーターとして使用されます。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、空間参照系の数値 ID は変更されません。

このパラメーターのデータ・タイプは INTEGER です。

x_offset

この空間参照系で表される形状の、すべての X 座標のオフセットを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、空間参照系の定義内のこのパラメーターの値は変更されません。

形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール係数 *x_scale*を適用する前にオフセットが引かれます。 (*WKT* は事前割り当てテキストであり、*WKB* は事前割り当てバイナリーです。)

このパラメーターのデータ・タイプは DOUBLE です。

x_scale

この空間参照系で表される形状の、すべての X 座標のスケール係数を指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、空間参照系の定義内のこのパラメーターの値は変更されません。

形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *x_offset* が引かれた後で、スケール係数が適用されます (乗算)。

このパラメーターのデータ・タイプは DOUBLE です。

y_offset

この空間参照系で表される形状の、すべての Y 座標のオフセットを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にする

ことができます。このパラメーターが NULL の場合、空間参照系の定義内のこのパラメーターの値は変更されません。

形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール係数 *y_scale*を適用する前にオフセットが引かれます。

このパラメーターのデータ・タイプは DOUBLE です。

y_scale
この空間参照系で表される形状の、すべての Y 座標のスケール係数を指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、空間参照系の定義内のこのパラメーターの値は変更されません。

形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *y_offset* が引かれた後で、スケール係数が適用されます (乗算)。 このスケール係数は *x_scale* と同じである必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

z_offset
この空間参照系で表される形状の、すべての Z 座標のオフセットを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、空間参照系の定義内のこのパラメーターの値は変更されません。

形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール係数 *z_scale*を適用する前にオフセットが引かれます。

このパラメーターのデータ・タイプは DOUBLE です。

z_scale
この空間参照系で表される形状の、すべての Z 座標のスケール係数を指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、空間参照系の定義内のこのパラメーターの値は変更されません。

形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *z_offset* が引かれた後で、スケール係数が適用されます (乗算)。

このパラメーターのデータ・タイプは DOUBLE です。

m_offset
この空間参照系で表される形状の、すべての M 座標のオフセットを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、空間参照系の定義内のこのパラメーターの値は変更されません。

形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール係数 *m_scale*を適用する前にオフセットが引かれます。

このパラメーターのデータ・タイプは DOUBLE です。

m_scale
この空間参照系で表される形状の、すべての M 座標のスケール係数を指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、空間参照系の定義内のこのパラメーターの値は変更されません。

形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *m_offset* が引かれた後で、スケール係数が適用されます (乗算)。

このパラメーターのデータ・タイプは DOUBLE です。

coordsys_name
この空間参照系の基礎となる座標システムを一意的に識別します。座標システムは、ビュー ST_COORDINATE_SYSTEMS にリストされる必要があります。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、この空間参照系に使用される座標システムは変更されません。

coordsys_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

description
アプリケーションを説明することにより、空間参照系を記述します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、空間参照系の記述は変更されません。

このパラメーターのデータ・タイプは VARCHAR(256) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_alter_srs ストアード・プロシージャーを呼び出す方法を示しています。 この例は、DB2 CALL コマンドを使用して、SRSDEMO という名前の空間参照系の *description* パラメーター値を変更します。

```
call db2gse.ST_alter_srs('SRSDEMO',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
     NULL,NULL,'SRS for GSE Demo Program: offices table',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

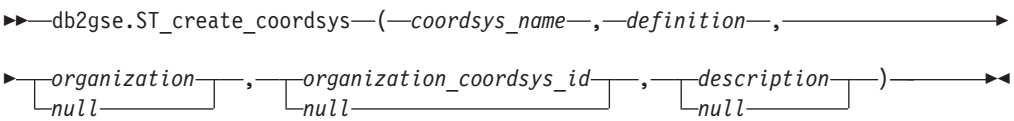
ST_create_coordsys

このストアード・プロシージャーは、新しい座標システムについての情報をデータベースに保管するために使用します。このストアード・プロシージャーが処理されると、座標システムについての情報が DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューに追加されます。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、DBADM 権限を持つ必要があります。

構文



パラメーターの説明

coordsys_name
座標システムを一意的に指定します。このパラメーターには NULL でない値を指定する必要があります。

coordsys_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

definition
座標システムを定義します。このパラメーターには NULL でない値を指定する必要があります。通常、座標システムを提供するベンダーがこのパラメーターの情報を提供します。

このパラメーターのデータ・タイプは VARCHAR(2048) です。

organization
座標系を定義し、その定義を提供した団体の名前。たとえば、「European

Petroleum Survey Group (EPSG)」など。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。

このパラメーターが NULL の場合、*organization_coordsys_id* パラメーターも NULL にする必要があります。このパラメーターが NULL でない場合、*organization_coordsys_id* パラメーターは NULL にできません。この場合、*organization* と *organization_coordsys_id* パラメーターを組み合わせて座標システムを一意的に識別します。

このパラメーターのデータ・タイプは VARCHAR(128) です。

organization_coordsys_id
数値 ID を指定します。*organization* パラメーターに指定された団体がこの値を割り当てます。この値は、すべての座標システムにまたがって一意的である必要はありません。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。

このパラメーターが NULL の場合、*organization* パラメーターも NULL にする必要があります。このパラメーターが NULL でない場合、*organization* パラメーターは NULL にできません。この場合、*organization* と *organization_coordsys_id* パラメーターを組み合わせて座標システムを一意的に識別します。

このパラメーターのデータ・タイプは INTEGER です。

description
アプリケーションを説明することにより、座標システムを記述します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、座標システムについての記述は記録されません。

このパラメーターのデータ・タイプは VARCHAR(256) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_create_coordsys ストアード・プロシージャーを呼び出す方法を示しています。 この例は DB2 CALL コマンドを使用して、次のパラメーター値を使用して座標システムを作成します。

- *coordsys_name* パラメーター: NORTH_AMERICAN_TEST
- *definition* パラメーター:

```
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]]
```

- *organization* パラメーター: EPSG
- *organization_coordsys_id* パラメーター: 1001
- *description* パラメーター: Test Coordinate Systems

```
call db2gse.ST_create_coordsys('NORTH_AMERICAN_TEST',
     'GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",
     SPHEROID["GRS_1980",6378137.0,298.257222101]],
     PRIMEM["Greenwich",0.0],UNIT["Degree",
     0.0174532925199433]]','EPSG',1001,'Test Coordinate Systems',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_create_srs

ストアード・プロシージャーは、空間参照系を作成するために使用します。

空間参照系は、座標システム、精度、およびこの空間参照系内で表現される座標の範囲により定義されます。範囲とは、X、Y、Z、および M 座標の可能な最小と最大の座標値です。

内部的には、DB2 Spatial Extender は座標値を正の整数として保管します。 したがって計算中の、丸めエラーによる影響 (浮動小数点演算の実際の値によって大きく変わる) を減らすことができます。また、空間操作のパフォーマンスも大幅に改善できます。

このストアード・プロシージャーには 2 つのバリエーションがあります。

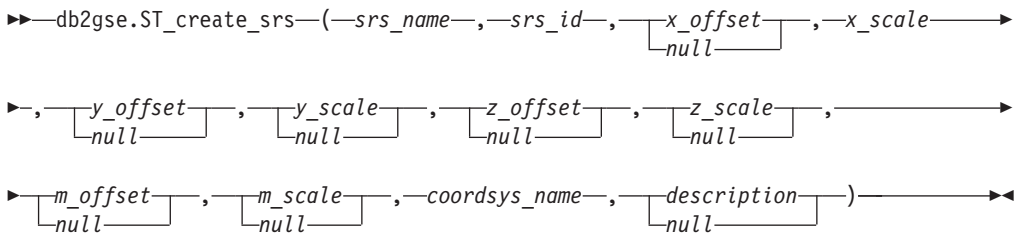
- 最初のバリエーションは、変換係数 (オフセットおよびスケール係数) を入力パラメーターとして取ります。
- 2 番目のバリエーションは、範囲と精度を入力パラメーターとして取り、変換係数を内部的に計算します。

許可

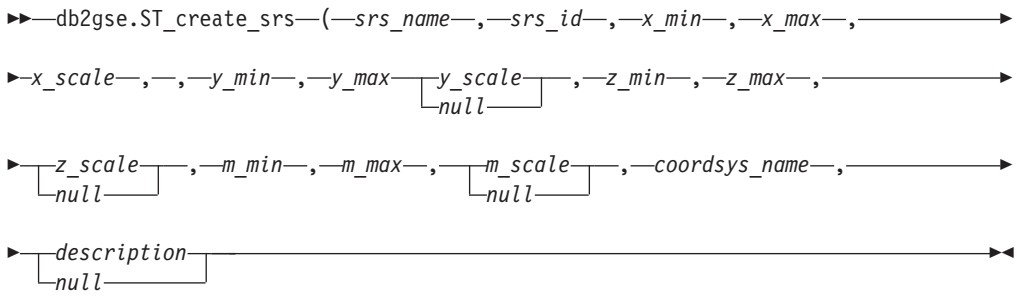
必要ありません。

構文

変換係数を使用する場合 (バージョン 1)



可能な最大範囲を使用する場合 (バージョン 2)



パラメーターの説明

変換係数を使用する場合 (バージョン 1)

srs_name
空間参照系を示します。このパラメーターには NULL でない値を指定する必要があります。

srs_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

srs_id
空間参照系を一意的に識別します。この数値 ID は、さまざまな空間処理関数の入力パラメーターとして使用されます。このパラメーターには NULL でない値を指定する必要があります。

測地参照系の場合、*srs_id* の値の範囲は、2000000318 から 2000001000 までででなければなりません。DB2 Geodetic Data Management Feature は、2000000000 から 2000000317 までの範囲の *srs_id* 値を持つ、定義済みの測地参照系を提供しています。

このパラメーターのデータ・タイプは INTEGER です。

x_offset
この空間参照系で表される形状の、すべての X 座標のオフセットを指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール係数 *x_scale*を適用する前にオフセットが引かれます。 (*WKT* は事前割り当てテキストであり、*WKB* は事前割り当てバイナリー

です。) このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、値 0 (ゼロ) が使用されます。

このパラメーターのデータ・タイプは DOUBLE です。

x_scale
この空間参照系で表される形状の、すべての X 座標のスケール係数を指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *x_offset* が引かれた後で、スケール係数が適用されます (乗算)。 *x_offset* 値を明示的に指定することも、デフォルトの *x_offset* 値 0 を使用することもできます。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

y_offset
この空間参照系で表される形状の、すべての Y 座標のオフセットを指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール係数 *y_scale*を適用する前にオフセットが引かれます。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL 値の場合、値 0 (ゼロ) が使用されます。

このパラメーターのデータ・タイプは DOUBLE です。

y_scale
この空間参照系で表される形状の、すべての Y 座標のスケール係数を指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *y_offset* が引かれた後で、スケール係数が適用されます (乗算)。 *y_offset* 値を明示的に指定することも、デフォルトの *y_offset* 値 0 を使用することもできます。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、*x_scale* パラメーターの値が使用されます。このパラメーターに NULL 以外の値を指定する場合、指定する値は *x_scale* パラメーターの値と一致する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

z_offset
この空間参照系で表される形状の、すべての Z 座標のオフセットを指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール係数 *z_scale*を適用する前にオフセットが引かれます。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、値 0 (ゼロ) が使用されます。

このパラメーターのデータ・タイプは DOUBLE です。

z_scale
この空間参照系で表される形状の、すべての Z 座標のスケール係数を指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *z_offset* が引かれた後で、スケール係数が適用されます (乗算)。 *z_offset* 値を明示的に指定することも、デフォルトの

z_offset 値 0 を使用することもできます。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、値 1 が使用されます。

このパラメーターのデータ・タイプは DOUBLE です。

m_offset
この空間参照系で表される形状の、すべての M 座標のオフセットを指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール係数 *m_scale*を適用する前にオフセットが引かれます。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、値 0 (ゼロ) が使用されます。

このパラメーターのデータ・タイプは DOUBLE です。

m_scale
この空間参照系で表される形状の、すべての M 座標のスケール係数を指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *m_offset* が引かれた後で、スケール係数が適用されます (乗算)。 *m_offset* 値を明示的に指定することも、デフォルトの *m_offset* 値 0 を使用することもできます。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、値 1 が使用されます。

このパラメーターのデータ・タイプは DOUBLE です。

coordsys_name
この空間参照系の基礎となる座標システムを一意的に識別します。座標システムは、ビュー ST_COORDINATE_SYSTEMS にリストされる必要があります。このパラメーターには NULL でない値を指定する必要があります。

coordsys_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

description
アプリケーションの目的を説明し、この空間参照系を説明します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、記述情報は記録されません。

このパラメーターのデータ・タイプは VARCHAR(256) です。

可能な最大範囲を使用する場合 (バージョン 2)

srs_name
空間参照系を示します。このパラメーターには NULL でない値を指定する必要があります。

srs_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

srs_id
空間参照系を一意的に識別します。この数値 ID は、さまざまな空間処理関数の入力パラメーターとして使用されます。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは INTEGER です。

x_min
この空間参照系を使用するすべての形状の、可能な最小の X 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

x_max
この空間参照系を使用するすべての形状の、可能な最大の X 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

x_scale の値によっては、ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS に表示される値は、ここで指定した値よりも大きい場合があります。ビューからの値が正しい値です。

このパラメーターのデータ・タイプは DOUBLE です。

x_scale
この空間参照系で表される形状の、すべての X 座標のスケール係数を指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *x_offset* が引かれた後で、スケール係数が適用されます (乗算)。 オフセット *x_offset* の計算は、*x_min* 値を基にします。このパラメーターには NULL でない値を指定する必要があります。

x_scale と *y_scale* の両方のパラメーターを指定する場合は、この両者の値は一致する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

y_min
この空間参照系を使用するすべての形状の、可能な最小の Y 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

y_max
この空間参照系を使用するすべての形状の、可能な最大の Y 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

y_scale の値によっては、ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS に表示される値は、ここで指定した値よりも大きい場合があります。ビューからの値が正しい値です。

このパラメーターのデータ・タイプは DOUBLE です。

y_scale
この空間参照系で表される形状の、すべての Y 座標のスケール係数を指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *y_offset* が引かれた後で、スケール係数が適用されます (乗算)。 オフセット *y_offset* の計算は、*y_min* 値を基にします。このパラメーターには値を指定する必要がありますが、値は NULL にすることが

できます。このパラメーターが NULL の場合、*x_scale* パラメーターの値が使用されます。*y_scale* と *x_scale* の両方のパラメーターを指定する場合は、この両者の値は一致する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

z_min
この空間参照系を使用するすべての形状の、可能な最小の Z 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

z_max
この空間参照系を使用するすべての形状の、可能な最大の Z 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

z_scale の値によっては、ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS に表示される値は、ここで指定した値よりも大きい場合があります。ビューからの値が正しい値です。

このパラメーターのデータ・タイプは DOUBLE です。

z_scale
この空間参照系で表される形状の、すべての Z 座標のスケール係数を指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *z_offset* が引かれた後で、スケール係数が適用されます (乗算)。 オフセット *z_offset* の計算は、*z_min* 値を基にします。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、値 1 が使用されます。

このパラメーターのデータ・タイプは DOUBLE です。

m_min
この空間参照系を使用するすべての形状の、可能な最小の M 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

m_max
この空間参照系を使用するすべての形状の、可能な最大の M 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

m_scale の値によっては、ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS に表示される値は、ここで指定した値よりも大きい場合があります。ビューからの値が正しい値です。

このパラメーターのデータ・タイプは DOUBLE です。

m_scale
この空間参照系で表される形状の、すべての M 座標のスケール係数を指定します。形状が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、オフセット *m_offset* が引かれた後で、スケール係数が適用されます (乗算)。 オフセット *m_offset* の計算は、*m_min* 値を基にします。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、値 1 が使用されます。

このパラメーターのデータ・タイプは DOUBLE です。

coordsys_name
この空間参照系の基礎となる座標システムを一意的に識別します。座標システムは、ビュー ST_COORDINATE_SYSTEMS にリストされる必要があります。このパラメーターには NULL でない値を指定する必要があります。

coordsys_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

description
アプリケーションの目的を説明し、この空間参照系を説明します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、記述情報は記録されません。

このパラメーターのデータ・タイプは VARCHAR(256) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_create_srs ストアード・プロシージャーを呼び出す方法を示しています。 この例は DB2 CALL コマンドを使用し、次のパラメーター値を使用して、SRSDEMO という名前の空間参照系を作成します。

- *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',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_disable_autogeocoding

このストアード・プロシージャーは、DB2 Spatial Extender が、ジオコーディングされた列とこの列に関連するジオコーディングする列との同期化を停止することを指定するために使用します。

ジオコーディング列 は、ジオコーダーへの入力として使用されます。

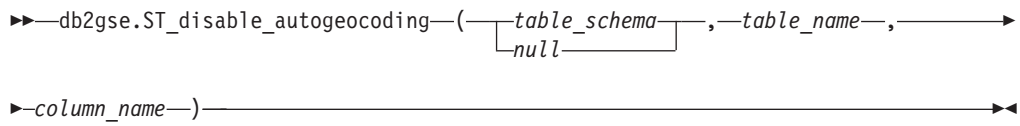
許可

このストアード・プロシージャーを呼び出すユーザー ID は、次の権限または特権の 1 つを持つ必要があります。

- ドロップされるトリガーが定義されている表を含むデータベースに関する、DBADM 権限 および DATAACCESS 権限。
- この表に対する CONTROL 特権
- この表に対する ALTER 特権または UPDATE 特権

注 : CONTROL および ALTER 特権の場合、DB2GSE スキーマに関する DROPIN 権限をもっている必要があります。

構文



パラメーターの説明

table_schema

table_name パラメーターに指定された表またはビューが属するスキーマの名前。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、表またはビューのスキーマ名として、CURRENT SCHEMA 特殊レジスター内の値が使用されます。

table_schema 値は、引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

table_name

ドロップするトリガーが定義された表の、修飾しない名前を指定します。このパラメーターには NULL でない値を指定する必要があります。

table_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

column_name
ドロップしようとするトリガーにより保守されている、ジオコーディングされた列の名前。このパラメーターには NULL でない値を指定する必要があります。

column_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_disable_autogeocoding ストアード・プロシージャーを呼び出す方法を示しています。 この例は、DB2 CALL コマンドを使用して、CUSTOMERS という名前の表の LOCATION 列の自動ジオコーディングを使用不可にしています。

```
call db2gse.ST_disable_autogeocoding(NULL,'CUSTOMERS','LOCATION',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_disable_db

このストアード・プロシージャーは、DB2 Spatial Extender が空間データを保管およびサポートするために必要なリソースの除去に使用します。

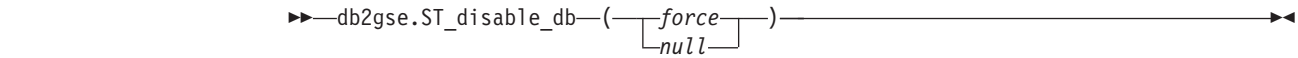
このストアード・プロシージャーは、データベースを空間操作に使用できるようにした後で問題などが起こった場合に、その解決に役立ちます。例えば、データベースを空間操作に使用できるようにした後で、このデータベースではなく別のデータベースを DB2 Spatial Extender で使用することを決めたとします。 まだ空間列を何も定義していない場合、または空間データを何もインポートしていない場合には、このストアード・プロシージャーを呼び出して、最初のデータベースからすべての空間リソースを除去することができます。空間列とそのタイプ定義の間には相互に

依存関係があるため、これらのタイプの列が存在する場合は、タイプ定義をドロップできません。すでに空間列を定義したデータベースの空間操作を使用不可にしようとする場合は、*force* パラメーターに 0 (ゼロ) 以外の値を指定して、データベース内のすべての空間リソース (他の依存関係を持たないもの) を除去する必要があります。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、DB2 Spatial Extender のリソースを除去するデータベースに対して、DBADM 権限を持つ必要があります。

構文



パラメーターの説明

force

空間タイプまたは空間処理関数に依存するデータベース・オブジェクトが存在したとしても、データベースを空間操作に使用できないようにすることを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。*force* パラメーターに 0 (ゼロ) または NULL 以外の値を指定すると、データベースは使用不可になり、DB2 Spatial Extender のすべてのリソースは除去されます (可能ならば)。 0 (ゼロ) または NULL を指定すると、いずれかのデータベース・オブジェクトが空間タイプまたは空間処理関数に依存する場合には、データベースは使用不可にされません。このような従属関係を持つ可能性のあるデータベース・オブジェクトには、表、ビュー、制約、トリガー、生成列、メソッド、関数、プロシージャー、およびその他のデータ・タイプ (空間属性を持つサブタイプまたは構造化タイプ) が含まれます。

このパラメーターのデータ・タイプは SMALLINT です。

出力パラメーター

msg_code

ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text

ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_disable_db ストアード・プロシージャーを呼び出す方法を示しています。 この例は DB2 CALL コマンドを使用して、*force* パラメーターに値 1 を使用し、データベースの空間操作を使用不可にします。

```
call db2gse.ST_disable_db(1,?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_drop_coordsys

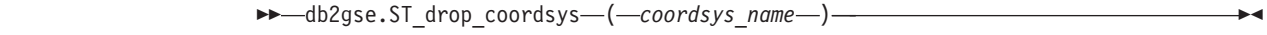
このストアード・プロシージャーは、データベースから座標システムについての情報を削除するために使用します。このストアード・プロシージャーが処理されると、座標システムについての情報が DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューから除去されます。

制約事項:
空間参照系の基になっている座標システムをドロップすることはできません。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、DBADM 権限を持つ必要があります。

構文



パラメーターの説明

coordsys_name
座標システムを一意的に指定します。このパラメーターには NULL でない値を指定する必要があります。

coordsys_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_drop_coordsys ストアード・プロシージャーを呼び出す方法を示しています。 この例は、DB2 CALL コマンドを使用して、NORTH_AMERICAN_TEST という名前の座標システムをデータベースから削除します。

```
call db2gse.ST_drop_coordsys('NORTH_AMERICAN_TEST',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_drop_srs

このストアード・プロシージャーは、空間参照系をドロップするために使用します。

このストアード・プロシージャーが処理されると、空間参照系についての情報は DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビューから除去されます。

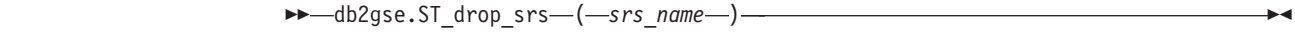
制約事項: その空間参照系を使用する空間列が登録されている場合は、その空間参照系をドロップすることはできません。

重要:
このストアード・プロシージャーを使用する場合は注意が必要です。このストアード・プロシージャーを使用して空間参照系をドロップし、その空間参照系に関連付けられた空間データがある場合には、その空間データに対する空間操作は実行できなくなります。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、DBADM 権限を持つ必要があります。

構文



パラメーターの説明

srs_name
空間参照系を示します。このパラメーターには NULL でない値を指定する必要があります。

srs_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_drop_srs ストアード・プロシージャーを呼び出す方法を示しています。 この例は、DB2 CALL コマンドを使用して、SRSDEMO という名前の空間参照系を削除します。

```
call db2gse.ST_drop_srs('SRSDEMO',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_enable_autogeocoding

このストアード・プロシージャーは、DB2 Spatial Extender が、ジオコーディングされた列とこの列に関連するジオコーディングされる列とを同期化するように指定するために使用します。

ジオコーディング列 は、ジオコーダーへの入力として使用されます。ジオコーディング列に値が挿入または更新されるたびに、トリガーがアクティブ化されます。これらのトリガーは、関連付けられたジオコーダーを呼び出して、挿入または更新された値をジオコーディングし、結果のデータをジオコーディングされた列に入れます。

制約事項: 自動ジオコーディングを使用可能にできるのは、INSERT および UPDATE トリガーを作成できる表だけです。したがって、ビューやニックネームに自動ジオコーディングを使用可能にすることはできません。

前提条件: 自動ジオコーディングを使用可能にする前に、ST_setup_geocoding ストアード・プロシージャーを呼び出して、ジオコーディングのセットアップ・ステップを実行する必要があります。ジオコーディング・セットアップ・ステップは、ジオコーダーおよびジオコーディングのパラメーター値を指定します。また、ジオコーディング列と同期化する相手のジオコーディングされる列も指定します。

許可

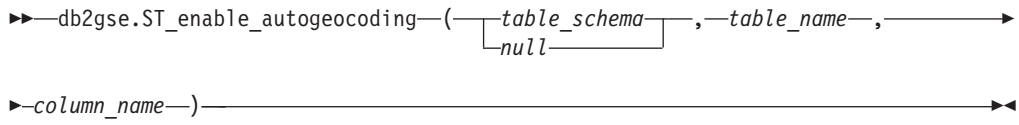
このストアード・プロシージャーを呼び出すユーザー ID は、次の権限または特権の 1 つを持つ必要があります。

- このストアード・プロシージャーにより作成されるトリガーが定義されている表を含むデータベースに対する、DBADM 権限。
- 表に対する CONTROL 特権
- 表に対する ALTER 特権

ステートメントの許可 ID が DBADM 権限を持たない場合、ステートメントの許可 ID が持つ特権 (PUBLIC またはグループ特権を考慮せずに) は、トリガーが存在するかぎり、次のすべての特権を含む必要があります。

- 自動ジオコーディングを使用可能にする表に対する SELECT 特権 または DATAACCESS 権限
- ジオコーディング・セットアップ内のパラメーターに指定された SQL 式を評価するために必要な特権。

構文



パラメーターの説明

table_schema

table_name パラメーターに指定された表が属するスキーマの名前。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、表のスキーマ名として、CURRENT SCHEMA 特殊レジスター内の値が使用されます。

table_schema 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

table_name

ジオコーディングされたデータが挿入または更新される列を含む表の、修飾されていない名前。このパラメーターには NULL でない値を指定する必要があります。

table_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

column_name
ジオコーディングされたデータが挿入または更新される列の名前。この列は、ジオコーディングされた列と呼ばれます。このパラメーターには NULL でない値を指定する必要があります。

column_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_enable_autogeocoding ストアード・プロシージャーを呼び出す方法を示しています。この例は、DB2 CALL コマンドを使用して、CUSTOMERS という名前の表の LOCATION 列の自動ジオコーディングを使用可能にしています。

```
call db2gse.ST_enable_autogeocoding(NULL,'CUSTOMERS','LOCATION',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_enable_db

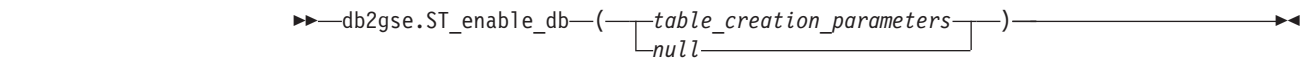
このストアード・プロシージャーは、空間データを保管し、空間操作をサポートするのに必要なリソースを、データベースに提供するために使用します。これらのリソースには、空間データ、空間インデックス・タイプ、カタログ・ビュー、提供された関数、およびその他のストアード・プロシージャーが含まれます。

このストアード・プロシージャーは、db2gse.gse_enable_db を置き換えます。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、使用可能にするデータベースに対して DBADM 権限を持つ必要があります。

構文



パラメーターの説明

table_creation_parameters

DB2 Spatial Extender カタログ表の CREATE TABLE ステートメントに追加する、オプションを指定します。 このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、CREATE TABLE ステートメントにオプションは追加されません。

これらのオプションを指定するには、DB2 CREATE TABLE ステートメントの構文を使用します。 例えば、作成しようとする表のための表スペースを指定するには、次の構文を使用します。

```
IN tsName INDEX IN indexTsName
```

このパラメーターのデータ・タイプは VARCHAR(32K) です。

出力パラメーター

msg_code

ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text

ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

次の例は、CLI (コール・レベル・インターフェース) を使用して、ST_enable_db ストアード・プロシージャーを呼び出す方法を示しています。

```
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;

/* Allocate environment handle */
rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);

/* Allocate database handle */
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);


/* Allocate statement handle */
rc = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt) ;

/* Associate SQL statement to call the ST_enable_db stored procedure    */
/* with statement handle and send the statement to DBMS to be prepared. */
rc = SQLPrepare(hstmt, "call db2gse!ST_enable_db(?,?,?)", SQL_NTS);

/* Bind 1st parameter marker in the SQL call statement, the input */
/* parameter for table creation parameters, to variable           */
/* table_creation_parameters.                                     */
ind[0] = SQL_NULL_DATA;
rc = SQLBindParameter(hstmt, 1, SQL_PARAM_OUTPUT, SQL_C_CHAR,
     SQL_VARCHAR, 255, 0, table_creation_parameters, 256, &ind[0]);

/* Bind 2nd parameter marker in the SQL call statement, the output */
/* parameter for returned message code, to variable msg_code.      */
ind[1] = 0;
rc = SQLBindParameter(hstmt, 2, SQL_PARAM_OUTPUT, SQL_C_LONG,
                      SQL_INTEGER, 0, 0, &msg_code, 4, &ind[1]);

/* Bind 3rd parameter marker in the SQL call statement, the output */
/* parameter returned message text, to variable msg_text.          */
ind[2] = 0;
rc = SQLBindParameter(hstmt, 3, SQL_PARAM_OUTPUT, SQL_C_CHAR,
                      SQL_VARCHAR, (sizeof(msg_text)-1), 0, msg_text,
                       sizeof(msg_text), &ind[2]);
rc = SQLExecute(hstmt);
```

ST_export_shape

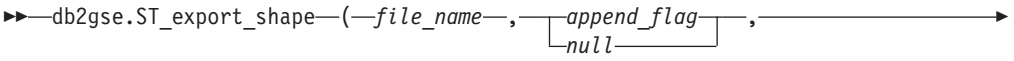
このストアード・プロシージャーは、空間列とこの列に関連する表を形状ファイルにエクスポートするために使用します。

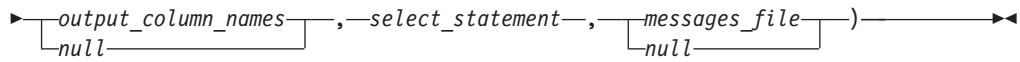
許可

このストアード・プロシージャーを呼び出すユーザー ID は、データをエクスポートするための SELECT ステートメントを正常に実行するために必要な特権を持つ必要があります。

ストアード・プロシージャーは、DB2 インスタンス所有者が所有するプロセスとして実行され、形状ファイルを作成または書き込むために必要な特権を、サーバー・マシン上でもつ必要があります。

構文





パラメーターの説明

file_name
指定されたデータのエクスポート先である形状ファイルの完全なパス名を指定します。このパラメーターには NULL でない値を指定する必要があります。

ST_export_shape ストアード・プロシージャーを使用して、新しいファイルにエクスポートしたり、エクスポートされるデータを付加する形で、既存のファイルにエクスポートすることができます。

- 新しいファイルにエクスポートする場合、オプションのファイル拡張子として .shp または .SHP を指定することができます。ファイル拡張子として .shp または .SHP を指定すると、DB2 Spatial Extender は指定された *file_name* 値を使用してファイルを作成します。オプションのファイル拡張子を指定しないと、DB2 Spatial Extender は、指定した *file_name* 値と拡張子 .shp の名前を持つファイルを作成します。
- 既存ファイルに付加する形でデータをエクスポートする場合、DB2 Spatial Extender は最初に、*file_name* パラメーターに指定された名前と正確に一致する名前を探します。正確に一致する名前を DB2 Spatial Extender が見つけられない場合は、まず .shp 拡張子を持つファイルを探し、次に .SHP 拡張子を持つファイルを探します。

append_flag パラメーターの値が、既存ファイルへの付加ではないことを示しているが、*file_name* パラメーターに指定された名前のファイルが既に存在する場合、DB2 Spatial Extender はエラーを戻し、ファイルの上書きはしません。

サーバー・マシンに書き込まれるファイルのリストについては、『使用上の注意』を参照してください。ストアード・プロシージャーは、DB2 インスタンス所有者が所有するプロセスとして実行され、ファイルを作成または書き込むために必要な特権を、サーバー・マシン上でもつ必要があります。

このパラメーターのデータ・タイプは VARCHAR(256) です。

append_flag
エクスポートされるデータを既存の形状ファイルに付加するかどうかを示します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。既存の形状ファイルに付加するかどうかを、次のように指定します。

- 既存の形状ファイルにデータを付加する場合は、0 (ゼロ) および NULL 以外の任意の値を指定します。この場合、ファイルの構造はエクスポートされるデータと一致する必要があり、一致していないとエラーが戻されます。
- 新しいファイルにエクスポートする場合は、0 (ゼロ) または NULL を指定します。この場合 DB2 Spatial Extender は、既存ファイルは一切上書きしません。

このパラメーターのデータ・タイプは SMALLINT です。

output_column_names
出力 dBASE ファイル内の、空間以外の列に使用される 1 つまたは複数の列名 (コンマで区切る) を指定します。このパラメーターには値を指定する必要があ

りますが、値は NULL にすることができます。このパラメーターが NULL の場合、SELECT ステートメントからの名前が使用されます。

このパラメーターを指定し、名前を二重引用符で囲まないと、列名は英大文字に変換されます。指定する列の数は、空間列を除き、*select_statement* パラメーターで指定された、SELECT ステートメントから戻される列の数と一致する必要があります。

このパラメーターのデータ・タイプは VARCHAR(32K) です。

select_statement
エクスポートされるデータを戻す副選択を指定します。副選択は、ただ 1 つの空間列および、任意の数の属性列を参照する必要があります。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは VARCHAR(32K) です。

messages_file
エクスポート操作についてのメッセージを含む、(サーバー・マシン上の) ファイルの完全なパス名を指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、DB2 Spatial Extender メッセージ用のファイルは作成されません。

このメッセージ・ファイルに送られるメッセージには、次のものがあります。

- エクスポート操作のサマリーなどの、通知メッセージ
- エクスポートできなかったデータのエラー・メッセージ (例えば、座標システムが異なるなどの理由から)

ストアード・プロシージャーは、DB2 インスタンス所有者が所有するプロセスとして実行され、ファイルを作成するために必要な特権をサーバー・マシン上でもつ必要があります。

このパラメーターのデータ・タイプは VARCHAR(256) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

使用上の注意

一度にエクスポートできる空間列は 1 つだけです。

ST_export_shape ストアード・プロシージャーは次の 4 つのファイルを作成または書き込みます。

- メインの形状ファイル (.shp 拡張子)。
- 形状索引ファイル (.shx 拡張子)。
- 空間以外の列のデータを含む dBASE ファイル (.dbf 拡張子)。このファイルは、属性列を実際にエクスポートする必要がある場合のみ作成されます。
- 空間データに関連した座標系を指定する展開ファイル (座標系が ″UNSPECIFIED″ と等しくない場合) (.prj 拡張子)。座標システムは、最初の空間レコードから得ることができます。後続のレコードが異なる座標システムを持つ場合、エラーが起こります。

次の表は、DB2 のデータ・タイプがどのように dBASE 属性ファイルに保管されるかを示しています。 その他の DB2 のデータ・タイプはすべて、サポートされません。

表 27. 属性ファイル内での DB2 のデータ・タイプの保管

SQL タイプ	.dbf タイプ	.dbf の長さ	.dbf の小数部	コメント
SMALLINT	N	6	0	
INTEGER	N	11	0	
BIGINT	N	20	0	
DECIMAL	N	precision+2	scale	
REAL FLOAT(1) から FLOAT(24)	F	14	6	
DOUBLE FLOAT(25) から FLOAT(53)	F	19	9	
CHARACTER、VARCHAR、 LONG VARCHAR、および DATALINK	C	*len*	0	length ≤ 255
DATE	D	8	0	
TIME	C	8	0	
TIMESTAMP	C	26	0	

上の表にリストされたタイプに基づく、データ・タイプおよび特殊タイプのシノニムは、すべてサポートされます。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_export_shape ストアード・プロシージャーを呼び出す方法を示しています。 この例は DB2 CALL コマンドを使用して、CUSTOMERS 表からすべての行を形状ファイルにエクスポートします。この形状ファイルは作成され、/tmp/export_file と名前が付けられます。

```
call db2gse.ST_export_shape('/tmp/export_file',0,NULL,
     'select * from customers','/tmp/export_msg',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_import_shape

このストアード・プロシージャーは、空間操作が使用可能になっているデータベースに、形状ファイルをインポートするために使用します。

ストアード・プロシージャーは *create_table_flag* パラメーターに基づき、次の 2 つの方法のいずれかで操作することができます。

- DB2 Spatial Extender は、1 つの空間列といくつかの属性列を持つ表を作成してから、この表の列にファイルのデータをロードすることができます。
- 上記以外の場合、形状および属性のデータは、ファイルのデータと一致する空間列と属性列を持つ既存の表にロードすることができます。

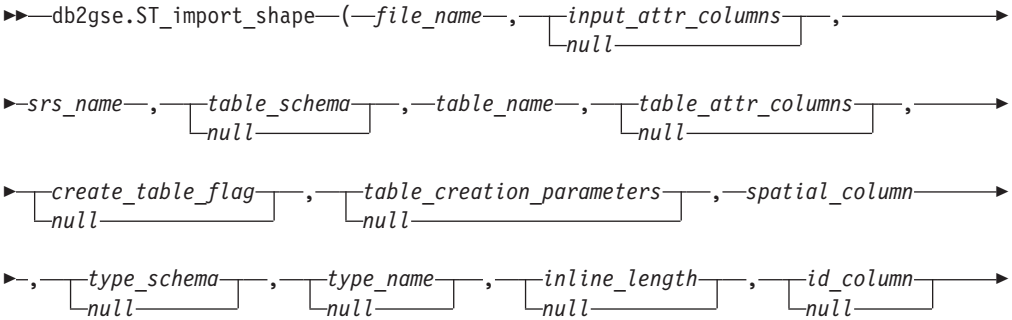
許可

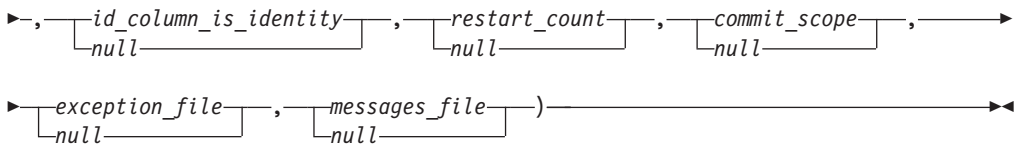
DB2 インスタンスの所有者は、入力ファイルを読み取り、またオプションとしてエラー・ファイルを書き込むために必要な特権を、サーバー・マシン上でもつ必要があります。 追加の許可要件は、既存の表にインポートするのか、新しい表にインポートするのかにより異なります。

- **既存の表にインポートする場合**、このストアード・プロシージャーを呼び出すユーザー ID は、次の権限または特権の 1 つを持つ必要があります。
  - DATAACCESS
  - 表またはビューに関する CONTROL 特権
  - 表またはビューに対する INSERT 特権と SELECT 特権
- **新しい表にインポートする場合**、このストアード・プロシージャーを呼び出すユーザー ID は、次の権限または特権の 1 つを持つ必要があります。
  - DBADM
  - データベースに対する CREATETAB 権限

  ユーザー ID には次の権限の 1 つも必要です。
  - 表のスキーマ名が存在しない場合は、データベースに対する IMPLICIT_SCHEMA 権限。
  - 表のスキーマが存在する場合は、スキーマに対する CREATEIN 特権。

構文





パラメーターの説明

file_name

インポートされる形状ファイルの完全なパス名を指定します。このパラメーターには NULL でない値を指定する必要があります。

オプションのファイル拡張子を指定する場合は、.shp または .SHP のいずれかを指定します。DB2 Spatial Extender はまず、指定されたファイル名と完全に一致するものを探します。 正確に一致する名前を DB2 Spatial Extender が見つけられない場合は、まず .shp 拡張子を持つファイルを探し、次に .SHP 拡張子を持つファイルを探します。

サーバー・マシンに存在する必要がある、必須ファイルのリストについては、『使用上の注意』を参照してください。ストアード・プロシージャーは、DB2 インスタンス所有者が所有するプロセスとして実行され、ファイルを読むために必要な特権をサーバー・マシン上でもつ必要があります。

このパラメーターのデータ・タイプは VARCHAR(256) です。

input_attr_columns

dBASE ファイルからインポートする属性列のリストを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、すべての列がインポートされます。dBASE ファイルが存在しない場合、このパラメーターは空のストリングまたは NULL にする必要があります。

このパラメーターに NULL でない値を指定するには、次の指定の 1 つを使用します。

- **属性列の名前をリストする。** 次の例は、dBASE ファイルからインポートされる属性列の名前のリストを指定する方法を示しています。

  ```
  N(COLUMN1,COLUMN5,COLUMN3,COLUMN7)
  ```

  列名を二重引用符で囲まないと、英大文字に変換されます。リスト内のそれぞれの名前はコンマで区切る必要があります。結果の名前は、dBASE ファイル内の列名と正確に一致する必要があります。
- **属性列の番号をリストする。** 次の例は、dBASE ファイルからインポートされる属性列の番号のリストを指定する方法を示しています。

  ```
  P(1,5,3,7)
  ```

  列は 1 から始まる番号が振られています。リスト内のそれぞれの番号はコンマで区切る必要があります。
- **属性データをインポートしないことを示す。** 空のストリングである "" を指定すると、DB2 Spatial Extender が属性データを 1 つもインポートしない ことを明示的に指定することになります。

このパラメーターのデータ・タイプは VARCHAR(32K) です。

srs_name
空間列にインポートされる形状に使用される、空間参照系を指定します。このパラメーターには NULL でない値を指定する必要があります。

空間列は登録されません。空間参照系 (SRS) は、データをインポートする前に存在している必要があります。インポート処理は、暗黙に SRS を作成することはしませんが、.prj ファイル (形状ファイルで使用可能な場合) に指定された座標システムと SRS の座標システムとを比較します。またインポート処理は、形状ファイル内のデータの範囲が、与えられた空間参照系内で表現できるかを検査します。つまりインポート処理は、SRS の可能な最小および最大の X、Y、Z、および M 座標内に、範囲が収まるかどうかを検査します。

srs_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

table_schema
table_name パラメーターに指定された表が属するスキーマの名前。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、表またはビューのスキーマ名として、CURRENT SCHEMA 特殊レジスター内の値が使用されます。

table_schema 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

table_name
インポートされる形状ファイルをロードする表の、修飾されていない名前。このパラメーターには NULL でない値を指定する必要があります。

table_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

table_attr_columns
dBASE ファイルからの属性データを保管する表の列名。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、dBASE ファイル内の列名が使用されます。

このパラメーターを指定する場合、名前の番号は、dBASE ファイルからインポートされた列の番号と一致する必要があります。表が存在する場合、列の定義は、入ってくるデータと一致する必要があります。属性データのタイプが DB2 のデータ・タイプにどのようにマップされるかについては、『使用上の注意』を参照してください。

このパラメーターのデータ・タイプは VARCHAR(32K) です。

create_table_flag
インポート処理で新しい表を作成するかどうかを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL または 0 (ゼロ) 以外の値の場合、新しい表が作成されます。(表が既に存在する場合は、エラーになります。) このパラメーターが 0 (ゼロ) の場合、表は作成されず、表は既に存在している必要があります。

このパラメーターのデータ・タイプは INTEGER です。

table_creation_parameters
データのインポート先の表を作成する CREATE TABLE ステートメントに追加する、任意のオプションを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、CREATE TABLE ステートメントにオプションは追加されません。

何らかの CREATE TABLE オプションを指定するには、DB2 CREATE TABLE ステートメントの構文を使用します。 例えば、作成する表のための表スペースを指定するには、次の構文を使用します。

```
IN tsName INDEX IN indexTsName LONG IN longTsName
```

このパラメーターのデータ・タイプは VARCHAR(32K) です。

spatial_column
形状データのロード先の表の、空間列の名前。このパラメーターには NULL でない値を指定する必要があります。

新しい表の場合、このパラメーターは、作成される新しい空間列の名前を指定します。それ以外の場合、このパラメーターは表の中の既存の空間列の名前を指定します。

spatial_column 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

type_schema
新しい表に空間列を作成するときに使用される、空間データ (*type_name* パラメーターで指定したもの) のスキーマ名を指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、値 DB2GSE が使用されます。

type_schema 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

type_name
空間の値に使用されるデータ・タイプの名前。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、データ・タイプは形状ファイルにより判別され、それは次のタイプのうちの 1 つです。

- ST_Point
- ST_MultiPoint
- ST_MultiLineString
- ST_MultiPolygon

形状ファイルは、定義により、ポイントと複数ポイントの区別のみ可能であり、ポリゴンと複数ポリゴンの区別、または折れ線と複数折れ線の区別はできません。

まだ存在しない表にインポートする場合、このデータ・タイプは空間列のデータ・タイプにも使用されます。この場合、データ・タイプは、ST_Point、ST_MultiPoint、ST_MultiLineString、または ST_MultiPolygon のスーパータイプにすることもできます。

type_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

inline_length

新しい表について、表内の空間列に割り振ることができる最大バイト数を指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、CREATE TABLE ステートメント内で明示的な INLINE LENGTH オプションは使用されず、暗黙に DB2 のデフォルトが使用されます。

このサイズを超える空間レコードは LOB 表スペースに別に保管され、これのアクセスには時間がかかる可能性があります。

各種の空間タイプに必要な典型的なサイズは、次のとおりです。

- **1 つのポイント:** 292。
- **複数ポイント、折れ線、またはポリゴン:** できるだけ大きい値。1 行内の合計バイト数は、表が作成された表スペースのページ・サイズの限界を超えられないことに注意してください。

この値の完全な説明については、CREATE TABLE SQL ステートメントに関する DB2 の資料を参照してください。 また、既存の表に対するインライン形状の数および、インラインの長さを変更する機能を決定するには、db2dart ユーティリティーも参照してください。

このパラメーターのデータ・タイプは INTEGER です。

id_column

データの各行に対するユニークな番号を含めるために作成される列の名前。(ESRI ツールでは、列に SE_ROW_ID という名前を付ける必要があります。) この列のためのユニークな値は、インポート処理中に自動的に生成されます。このパラメーターには値を指定する必要がありますが、表の中に列 (各行にユニーク ID を持つ列) が存在しない場合、または新しく作成される表にこのような列を追加しない場合は、値を NULL にできます。このパラメーターが NULL の場合、ユニーク番号を持つ列は作成されず、またユニーク番号を入れることもしません。

制約事項: dBASE ファイル内の列名と一致する *id_column* 名を指定することはできません。

このパラメーターの要件と効果は、表が既に存在するかどうかにより、次のようになります。

- **既存の表の場合**、*id_column* パラメーターのデータ・タイプは、任意の整数タイプ (INTEGER、SMALLINT、または BIGINT) にできます。
- **新しい表が作成される場合**、ストアード・プロシージャーが表を作成するときに、列が表に追加されます。この列は次のように定義されます。

```
INTEGER NOT NULL PRIMARY KEY
```

id_column_is_identity パラメーターの値が NULL でなく、0 (ゼロ) でもない場合、定義は次のように展開されます。

```
INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY
( START WITH 1 INCREMENT BY 1 )
```

id_column 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

id_column_is_identity
指定された *id_column* が IDENTITY 節を使用して作成されるかどうかを示します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが 0 (ゼロ) または NULL の場合、列は ID 列としては作成されません。このパラメーターが 0 (ゼロ) または NULL 以外の値であれば、列は ID 列として作成されます。このパラメーターは、既存の表の場合は無視されます。

このパラメーターのデータ・タイプは SMALLINT です。

restart_count
インポート操作をレコード *n* + 1 から開始することを指定します。最初の *n* レコードはスキップされます。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、すべてのレコード (レコード番号 1 から開始) がインポートされます。

このパラメーターのデータ・タイプは INTEGER です。

commit_scope
少なくとも *n* レコードをインポートするたびに COMMIT を実行することを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、値 0 (ゼロ) が使用され、レコードはコミットされません。

このパラメーターのデータ・タイプは INTEGER です。

exception_file
インポートできなかった形状データを保管する、形状ファイルの完全なパス名を指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、ファイルは作成されません。

このパラメーターに値を指定し、オプションのファイル拡張子を含める場合は、.shp または .SHP のいずれかを指定します。拡張子が NULL の場合は、拡張子 .shp が付加されます。

この例外ファイルには、失敗した、1 つの INSERT ステートメントの操作対象となった行全体を含むブロックが入ります。例えば、形状データが誤ってエンコードされているために、1 つの行をインポートできなかったとします。1 つの INSERT ステートメントが、エラーの 1 行を含む 20 行をインポートしようとします。この場合、1 行だけが問題なのですが、20 行のブロック全体が例外ファイルに書き込まれます。

レコードが例外ファイルに書き込まれるのは、エラーのレコードを正しく識別できた場合 (例えば、形状レコード・タイプが無効である場合など) だけです。形状データ (.shp ファイル) および形状索引 (.shx ファイル) にある種の損傷が発

生すると、該当するレコードが識別できなくなります。この場合、例外ファイルにはレコードは書き込まれず、問題を報告するエラー・メッセージが出されます。

このパラメーターに値を指定すると、サーバー・マシンに 4 つのファイルが作成されます。これらのファイルの説明は、『使用上の注意』を参照してください。ストアード・プロシージャーは、DB2 インスタンス所有者が所有するプロセスとして実行され、ファイルを作成するために必要な特権をサーバー・マシン上でもつ必要があります。ファイルが既に存在する場合、ストアード・プロシージャーはエラーを戻します。

このパラメーターのデータ・タイプは VARCHAR(256) です。

messages_file
インポート操作についてのメッセージを含む、(サーバー・マシン上の) ファイルの完全なパス名を指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、DB2 Spatial Extender メッセージ用のファイルは作成されません。

メッセージ・ファイルに書き込まれるメッセージには、次のものがあります。

- インポート操作のサマリーなどの、通知メッセージ
- インポートできなかったデータのエラー・メッセージ (例えば、座標システムが異なるなどの理由から)

これらのメッセージは、例外ファイル (*exception_file* パラメーターで指定されたもの) に保管される形状データに対応しています。

ストアード・プロシージャーは、DB2 インスタンス所有者が所有するプロセスとして実行され、ファイルを作成するために必要な特権をサーバー・マシン上でもつ必要があります。ファイルが既に存在する場合、ストアード・プロシージャーはエラーを戻します。

このパラメーターのデータ・タイプは VARCHAR(256) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

使用上の注意

ST_import_shape ストアード・プロシージャーは、以下のファイルのうち、1 個から 4 個を使用します。

- メインの形状ファイル (.shp 拡張子)。このファイルは必須です。
- 形状索引ファイル (.shx 拡張子)。このファイルはオプションです。これが存在すれば、インポート操作のパフォーマンスを改善できます。
- 属性データを含む dBASE ファイル (.dbf 拡張子)。このファイルは、属性データをインポートする場合のみ必要です。
- 形状データの座標システムを指定する展開ファイル (.prj 拡張子)。このファイルはオプションです。このファイルが存在すると、この中で定義された座標システムが、*srs_id* パラメーターで指定された空間参照系の座標システムと比較されます。

次の表は、dBASE 属性データ・タイプと DB2 データ・タイプの対応を示しています。 その他の属性データ・タイプはすべて、サポートされません。

表 28. DB2 データ・タイプと dBASE 属性データ・タイプのリレーションシップ

.dbf タイプ	.dbf の長さb (注を参照)	.dbf の小数部 (注を参照)	SQL タイプ	コメント
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	

注: この表には次の変数が含まれ、両方とも dBASE ファイルのヘッダーに定義されています。

- *len* は、dBASE ファイル内の列の合計の長さを表します。DB2 Spatial Extender はこの値を次の 2 つの目的に使用します。
  - SQL データ・タイプ DECIMAL の精度、または SQL データ・タイプ CHAR の長さを定義するため
  - どの整数タイプまたは浮動小数点タイプを使用するかを決めるため
- *dec* は、dBASE ファイルにおいて、列の小数点の右側にある桁数の最大値を表します。 DB2 Spatial Extender はこの値を使用して、SQL データ・タイプ DECIMAL のスケールを定義します。

例えば、dBASE ファイルに 1 つのデータの列があり、その長さ (*len*) が 20 と定義されているとします。小数点の右の桁数 (*dec*) は 5 と定義されているとします。DB2 Spatial Extender はその列からデータをインポートするときに、*len* および *dec* の値を使用して、SQL データ・タイプ: DECIMAL(20,5) を導きます。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_import_shape ストアード・プロシージャーを呼び出す方法を示しています。 この例は DB2 CALL コマンドを使用して、/tmp/officesShape という名前の形状ファイルを OFFICES という名前の表にインポートします。

```
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',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_register_geocoder

このストアード・プロシージャーは、DB2SE_USA_GEOCODER ジオコーダー (DB2 Spatial Extender と一緒に配布されるもの) 以外のジオコーダーを登録するために使用します。 DB2SE_USA_GEOCODER ジオコーダーは、データベースを使用可能にすると DB2 Spatial Extender により登録されます。

前提条件: ジオコーダーを登録する前に、以下のことを行ってください。

- ジオコーダーをインプリメントする関数が既に作成されていることを確認します。各ジオコーダー関数は、一意的に識別されるジオコーダー名を持つジオコーダーとして登録することができます。
- ジオコーダーの提供会社から、次のような情報を入手します。
  - 関数を作成する SQL ステートメント
  - 形状データをサポートするために、ST_create_srs パラメーターに使用する値
  - 次のような、ジオコーダーを登録するための情報
    - ジオコーダーの説明
    - ジオコーダーのパラメーターの説明
    - ジオコーダー・パラメーターのデフォルト値

ジオコーダー関数の戻りタイプは、ジオコーディングされた列のデータ・タイプと一致する必要があります。ジオコーディング・パラメーターは、ジオコーダーが必要とするデータを含む列名 (*ジオコーディング列* と呼ばれる) にすることができます。例えば、ジオコーダー・パラメーターには、アドレスや、ジオコーダーにとって特別な意味を持つ値 (最小一致スコアなど) を指定できます。ジオコーディング・パラメーターが列名の場合、その列は、ジオコーディングされた列と同じ表またはビューにある必要があります。

ジオコーダー関数の戻りタイプは、ジオコーディングされた列のデータ・タイプとして働きます。この戻りタイプは、任意の DB2 データ・タイプ、ユーザー定義タイプ、または構造化タイプにすることができます。 ユーザー定義タイプまたは構造化タイプを戻す場合、ジオコーダー関数は該当のデータ・タイプにとって有効な値を戻す必要があります。ジオコーダー関数が ST_Geometry またはそのサブタイプの 1 つである、空間データ・タイプの値を戻す場合、ジオコーダー関数は有効な形状を作成する必要があります。形状は、既存の空間参照系を使用して表現する必要が

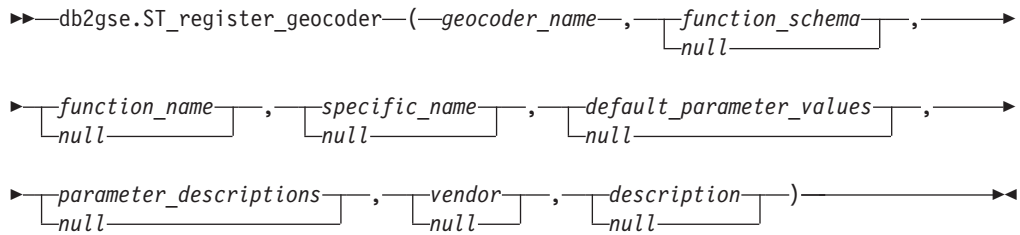
あります。形状に対して ST_IsValid 空間処理関数を呼び出した時に値 1 が戻されれば、その形状は有効です。ジオコーダー関数から戻されたデータは、ジオコーディング値を生成させた操作 (INSERT または UPDATE) により、ジオコーディングされた列に挿入またはそこで更新されます。

ジオコーダーが既に登録されているかどうかを調べるには、DB2GSE.ST_GEOCODERS カタログ・ビューを見てください。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、このストアード・プロシージャーが登録するジオコーダーを含むデータベースに対して、DBADM 権限を持つ必要があります。

構文



パラメーターの説明

geocoder_name
ジオコーダーを一意的に指定します。このパラメーターには NULL でない値を指定する必要があります。

geocoder_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

function_schema
このジオコーダーをインプリメントする関数のスキーマ名。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、関数のスキーマ名として、CURRENT SCHEMA 特殊レジスター内の値が使用されます。

function_schema 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

function_name
このジオコーダーをインプリメントする関数の、修飾されていない名前。関数は既に作成済みで、SYSCAT.ROUTINES にリストされている必要があります。

このパラメーターの場合、*specific_name* パラメーターが指定されていれば、NULL を指定することができます。*specific_name* パラメーターが指定されてい

ない場合、*function_name* 値は、暗黙的または明示的に定義された *function_schema* 値と合わせて、関数を一意的に特定できるものである必要があります。*function_name* パラメーターが指定されていない場合、DB2 Spatial Extender は SYSCAT.ROUTINES カタログ・ビューから *function_name* 値を検索します。

function_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

specific_name
ジオコーダーをインプリメントする関数の特定の名前を指定します。関数は既に作成済みで、SYSCAT.ROUTINES にリストされている必要があります。

このパラメーターの場合、*function_name* パラメーターが指定されており、*function_schema* と *function_name* を組み合わせてジオコーダー関数を一意的に識別できるならば、NULL を指定することができます。ジオコーダー関数名が多重定義 (次の記述を参照) の場合、*specific_name* パラメーターは NULL にはできません。(ある関数名が、1 つまたは複数の他の関数と同じであるが、パラメーターまたはパラメーターのデータ・タイプが同じではない場合、その関数名は*多重定義* となります。)

specific_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

default_parameter_values
ジオコーダー関数のデフォルトのジオコーディング・パラメーター値のリストを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。*default_parameter_values* パラメーター全体が NULL の場合、すべてのパラメーターのデフォルト値が NULL になります。

何らかのパラメーター値を指定する場合は、関数で定義された順序で、コンマで区切って指定します。例えば、以下のように指定します。

```
default_parm1_value,default_parm2_value,...
```

それぞれのパラメーター値は 1 つの SQL 式です。以下のガイドラインに従ってください。

- 値がストリングの場合は、単一引用符で囲みます。
- パラメーター値が数値の場合は、単一引用符で囲みません。
- パラメーター値が NULL の場合は、正しいタイプにキャストします。例えば、単に NULL と指定するのではなく、次のように指定します。

  ```
  CAST(NULL AS INTEGER)
  ```
- ジオコーディング・パラメーターがジオコーディングされる列になる場合は、デフォルトのパラメーター値を指定しないでください。

パラメーター値を指定しない場合 (つまり、2 つの連続するコンマを指定する (`...,,...`))、このパラメーターは、ジオコーディングがセットアップされる時

に指定するか、または該当のストアード・プロシージャーの *parameter_values* パラメーターを使用してバッチ・モードでジオコーディングを実行するときに、指定する必要があります。

このパラメーターのデータ・タイプは VARCHAR(32K) です。

parameter_descriptions
ジオコーダー関数のジオコーディング・パラメーター記述のリストを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。

parameter_descriptions パラメーター全体が NULL の場合、すべてのパラメーター記述が NULL になります。指定するそれぞれのパラメーター記述は、パラメーターの意味と使用法を説明するものであり、256 文字までの長さにできます。各パラメーターの記述は、コンマで区切り、関数で定義されたパラメーターの順序で指定する必要があります。パラメーターの記述内にコンマを使用する場合は、ストリングを単一引用符または二重引用符で囲みます。例えば、以下のように指定します。

```
description,'description2, which contains a comma',description3
```

このパラメーターのデータ・タイプは VARCHAR(32K) です。

vendor
ジオコーダーを作成したベンダー名。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、ジオコーダーを作成したベンダーの情報は記録されません。

このパラメーターのデータ・タイプは VARCHAR(128) です。

description
アプリケーションを説明することにより、ジオコーダーを記述します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、ジオコーダーについての記述情報は記録されません。

推奨事項: 次の情報を含めることをお勧めします。

- 事前割り当てテキスト (WKT) または事前割り当てバイナリー (WKB) のような空間データが戻される場合は、座標システムの名前
- ST_Geometry またはそのサブタイプが戻される場合は、空間参照系
- このジオコーダーが適用される地理上の地域名
- ユーザーが知っておくべき、ジオコーダーについてのその他の情報

このパラメーターのデータ・タイプは VARCHAR(256) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例では、入力として緯度と経度を取り、ST_Point 空間データ内にジオコーディングするジオコーダーを作成するとします。これを行うには、まず最初に lat_long_gc_func という名前の関数を作成します。次に、関数 lat_long_gc_func を使用する SAMPLEGC という名前のジオコーダーを登録します。

次に示すのは、ST_Point を戻す関数 lat_long_gc_func を作成する SQL ステートメントの例です。

```
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)
```

関数を作成した後、これをジオコーダーとして登録することができます。この例は、DB2 コマンド行プロセッサー CALL コマンドを使用して、ST_register_geocoder ストアード・プロシージャーを呼び出し、関数 lat_long_gc_func を使用する SAMPLEGC という名前のジオコーダーを登録する方法を示しています。

```
call db2gse.ST_register_geocoder ('SAMPLEGC',NULL,'LAT_LONG_GC_FUNC',',,1'
            ,NULL,'My Company','Latitude/Longitude to
            ST_Point Geocoder'?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_register_spatial_column

このストアード・プロシージャーは、空間列を登録し、この列と空間参照系 (SRS) を関連付けるために使用します。

このストアード・プロシージャーが処理されると、登録される空間列の情報が DB2GSE.ST_GEOMETRY_COLUMNS カタログ・ビューに追加されます。空間列を登録すると、可能な場合、すべての形状が指定された SRS を必ず使用するように、表に制約が作成されます。

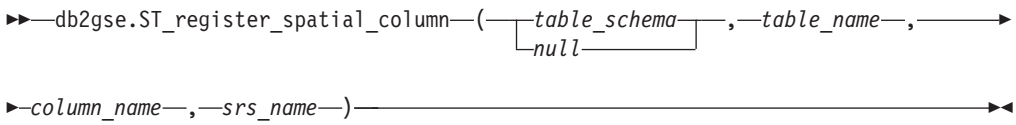
許可

このストアード・プロシージャーを呼び出すユーザー ID は、次の権限または特権の 1 つを持つ必要があります。

- 登録される空間列が属する表を含むデータベースに対する、DBADM 権限。
- この表に対する CONTROL 特権

• この表に対する ALTER 特権

構文



パラメーターの説明

table_schema
table_name パラメーターに指定された表またはビューが属するスキーマの名前。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、表またはビューのスキーマ名として、CURRENT SCHEMA 特殊レジスター内の値が使用されます。

table_schema 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

table_name
登録される列を含む表またはビューの、修飾されていない名前。このパラメーターには NULL でない値を指定する必要があります。

table_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

column_name
登録される列の名前。このパラメーターには NULL でない値を指定する必要があります。

column_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

srs_name
この空間列に使用される空間参照系の名前。このパラメーターには NULL でない値を指定する必要があります。

srs_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示

すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_register_spatial_column ストアード・プロシージャーを呼び出す方法を示しています。 この例は DB2 CALL コマンドを使用して、CUSTOMERS という名前の表の LOCATION という名前の空間列を登録しています。この CALL コマンドは *srs_name* パラメーター値に USA_SRS_1 を指定しています。

```
call db2gse.ST_register_spatial_column(NULL,'CUSTOMERS','LOCATION',
     'USA_SRS_1',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_remove_geocoding_setup

このストアード・プロシージャーは、ジオコーディングされた列に対するジオコーディング・セットアップ情報をすべて除去するために使用します。

このストアード・プロシージャーは、指定された「ジオコーディングされた列」に関連付けられた情報を、DB2GSE.ST_GEOCODING および DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビューから除去します。

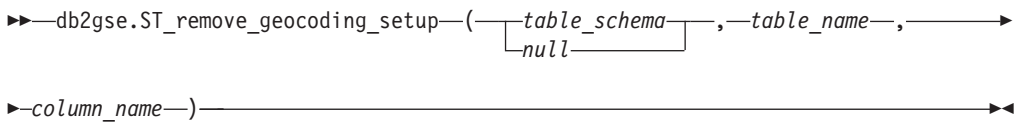
制約事項:
ジオコーディングされた列に対して自動ジオコーディングが使用可能になっている場合は、ジオコーディングのセットアップを除去できません。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、次の権限または特権の 1 つを持つ必要があります。

- 指定されたジオコーダーの操作対象の表が入っているデータベースに対する、DATAACCESS 権限。
- この表に対する CONTROL 特権
- この表に対する UPDATE 特権

構文



パラメーターの説明

table_schema

table_name パラメーターに指定された表またはビューが属するスキーマの名前。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、表またはビューのスキーマ名として、CURRENT SCHEMA 特殊レジスター内の値が使用されます。

table_schema 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

table_name

ジオコーディングされたデータが挿入または更新される列を含む表またはビューの、修飾されていない名前。このパラメーターには NULL でない値を指定する必要があります。

table_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

column_name

ジオコーディングされたデータが挿入または更新される列の名前。このパラメーターには NULL でない値を指定する必要があります。

column_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

出力パラメーター

msg_code

ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text

ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_remove_geocoding_setup ストアード・プロシージャーを呼び出す方法を示しています。 この例は DB2 CALL コマンドを使用して、CUSTOMER という名前の表の LOCATION 列のジオコーディング・セットアップを除去します。

```
call db2gse.ST_remove_geocoding_setup(NULL, 'CUSTOMERS', 'LOCATION',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_run_geocoding

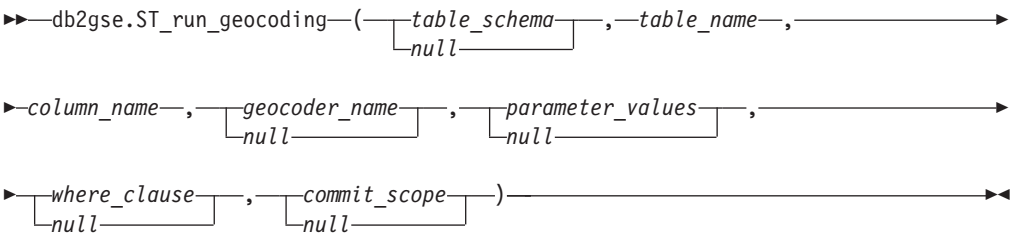
このストアード・プロシージャーは、ジオコーディングされる列に対して、ジオコーダーをバッチ・モードで実行するために使用します。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、次の権限または特権の 1 つを持つ必要があります。

- 指定されたジオコーダーの操作対象の表が入っているデータベースに対する、DATAACCESS 権限。
- この表に対する CONTROL 特権
- この表に対する UPDATE 特権

構文



パラメーターの説明

table_schema

table_name パラメーターに指定された表またはビューが属するスキーマの名前。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、表またはビューのスキーマ名として、CURRENT SCHEMA 特殊レジスター内の値が使用されます。

table_schema 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

table_name
ジオコーディングされたデータが挿入または更新される列を含む表またはビューの、修飾されていない名前。ビュー名を指定する場合、そのビューは更新可能なビューである必要があります。このパラメーターには NULL でない値を指定する必要があります。

table_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

column_name
ジオコーディングされたデータが挿入または更新される列の名前。このパラメーターには NULL でない値を指定する必要があります。

column_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

geocoder_name
ジオコーディングを実行するジオコーダーの名前。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、ジオコーディングのセットアップ時に指定されたジオコーダーにより、ジオコーディングが実行されます。

geocoder_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

parameter_values
ジオコーダー関数のジオコーディング・パラメーター値のリストを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。*parameter_values* パラメーター全体が NULL の場合、使用される値は、ジオコーダーのセットアップ時に指定されたパラメーター値または、ジオコーダーがセットアップされていない場合は、ジオコーダーのデフォルトのパラメーター値です。

何らかのパラメーター値を指定する場合は、関数で定義された順序で、コンマで区切って指定します。例えば、以下のように指定します。

```
parameter1-value,parameter2-value,...
```

各パラメーター値は、列名、ストリング、数値、または NULL にすることができます。

それぞれのパラメーター値は 1 つの SQL 式です。以下のガイドラインに従ってください。

- パラメーター値がジオコーディングされる列の名前である場合、その列は、ジオコーディングされた列と同じ表またはビューに存在する必要があります。
- パラメーター値がストリングの場合は、単一引用符で囲みます。
- パラメーター値が数値の場合は、単一引用符で囲みません。

- パラメーターが NULL の場合は、正しいタイプにキャストします。例えば、単に NULL と指定するのではなく、次のように指定します。

  ```
  CAST(NULL AS INTEGER)
  ```

パラメーター値を指定しない場合 (つまり、2 つの連続するコンマを指定する (...,,...))、このパラメーターは、ジオコーディングがセットアップされる時に指定するか、または該当のストアード・プロシージャーの *parameter_values* パラメーターを使用してバッチ・モードでジオコーディングを実行するときに、指定する必要があります。

このパラメーターのデータ・タイプは VARCHAR(32K) です。

where_clause

WHERE 節の本体を指定し、これにより、ジオコーディングされるレコードのセットに対する制約事項を定義します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。

where_clause パラメーターが NULL の場合、結果がどうなるかは、ストアード・プロシージャーを実行する前に、その列 (*column_name* パラメーターで指定された列) にジオコーディングがセットアップされているかどうかにより異なります。*where_clause* パラメーターが NULL であり、かつ、

- ジオコーディングのセットアップ時に値が指定された場合は、その値が *where_clause* パラメーターに使用されます。
- ジオコーディングがセットアップされていない、またはジオコーディングのセットアップ時に値が指定されなかった場合は、where 節は使用されません。

ジオコーダーの操作対象の表またはビューの、任意の列を参照する節を指定することができます。キーワード WHERE は指定しないでください。

このパラメーターのデータ・タイプは VARCHAR(32K) です。

commit_scope

n レコードをジオコーディングするたびに COMMIT を実行することを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。

commit_scope パラメーターが NULL の場合、結果がどうなるかは、ストアード・プロシージャーを実行する前に、その列 (*column_name* パラメーターで指定された列) にジオコーディングがセットアップされているかどうかにより異なります。*commit_scope* パラメーターが NULL であり、かつ、

- その列にジオコーディングをセットアップした時に値が指定された場合は、その値が *commit_scope* パラメーターに使用されます。
- ジオコーディングがセットアップされていない、またはセットアップされたが値が指定されていない場合、デフォルト値 0 (ゼロ) が使用され、これは COMMIT を実行しません。

このパラメーターのデータ・タイプは INTEGER です。

出力パラメーター

msg_code

ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、

または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_run_geocoding ストアード・プロシージャーを呼び出す方法を示しています。 この例は DB2 CALL コマンドを使用して、CUSTOMER という名前の表の LOCATION 列をジオコーディングします。 この CALL コマンドは、*geocoder_name* パラメーター値として DB2SE_USA_GEOCODER を、また *commit_scope* パラメーター値として 10 を指定しています。これにより、10 レコードをジオコーディングするたびに COMMIT が実行されます。

```
call db2gse.ST_run_geocoding(NULL, 'CUSTOMERS', 'LOCATION',
     'DB2SE_USA_GEOCODER',NULL,NULL,10,?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_setup_geocoding

このストアード・プロシージャーは、ジオコーディングする列をジオコーダーと関連付け、対応するジオコーディング・パラメーターをセットアップするために使用します。ここでセットアップされた情報は、DB2GSE.ST_GEOCODING および DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビューに記録されます。

このストアード・プロシージャーはジオコーディングを呼び出しません。これは、ジオコーディングされる列のパラメーター設定値を指定する手段を提供します。これらの設定値を使用すると、これ以後のバッチ・ジオコーディングまたは自動ジオコーディングの呼び出しは、より簡単なインターフェースにより行うことができます。このセットアップで指定したパラメーター設定値は、ジオコーダーの登録時に指定された、ジオコーダーのデフォルト・パラメーター値を変更します。また、バッチ・モードで ST_run_geocoding ストアード・プロシージャーを実行し、これらのパラメーター設定値を変更することもできます。

このステップは、自動ジオコーディングの前提条件となります。最初にジオコーディング・パラメーターを設定してからでないと、自動ジオコーディングを使用可能にすることはできません。このステップは、バッチ・ジオコーディングの場合は前提条件ではありません。バッチ・モードのジオコーディングは、セットアップ・ステップを実行してもしなくても、実行することができます。ただし、バッチ・ジオ

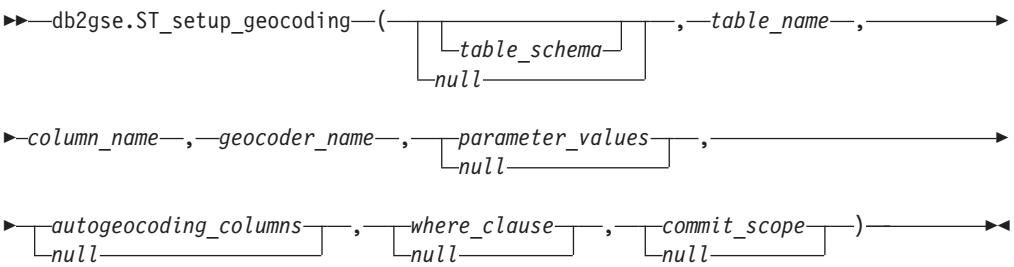
コーディングの前にセットアップ・ステップを実行しておけば、実行時に指定されていないパラメーター値はセットアップ時のものから取られます。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、次の権限または特権の 1 つを持つ必要があります。

- 指定されたジオコーダーの操作対象の表が入っているデータベースに対する、DATAACCESS 権限。
- この表に対する CONTROL 特権
- この表に対する UPDATE 特権

構文



パラメーターの説明

table_schema

table_name パラメーターに指定された表またはビューが属するスキーマの名前。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、表またはビューのスキーマ名として、CURRENT SCHEMA 特殊レジスター内の値が使用されます。

table_schema 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

table_name

ジオコーディングされたデータが挿入または更新される列を含む表またはビューの、修飾されていない名前。ビュー名を指定する場合、そのビューは更新可能なビューである必要があります。このパラメーターには NULL でない値を指定する必要があります。

table_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

column_name

ジオコーディングされたデータが挿入または更新される列の名前。このパラメーターには NULL でない値を指定する必要があります。

column_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

geocoder_name
ジオコーディングを実行するジオコーダーの名前。このパラメーターには NULL でない値を指定する必要があります。

geocoder_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

parameter_values
ジオコーダー関数のジオコーディング・パラメーター値のリストを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。*parameter_values* パラメーター全体が NULL の場合、使用される値は、ジオコーダーの登録時のデフォルトのパラメーター値です。

パラメーター値を指定する場合は、関数で定義された順序で、コンマで区切って指定します。例えば、以下のように指定します。

```
parameter1-value,parameter2-value,...
```

各パラメーター値は SQL 式であり、列名、ストリング、数値、または NULL にすることができます。以下のガイドラインに従ってください。

- パラメーター値がジオコーディングされる列の名前である場合、その列は、ジオコーディングされた列と同じ表またはビューに存在する必要があります。
- パラメーター値がストリングの場合は、単一引用符で囲みます。
- パラメーター値が数値の場合は、単一引用符で囲みません。
- パラメーター値を NULL 値として指定する場合は、正しいタイプにキャストします。例えば、単に NULL と指定するのではなく、次のように指定します。

  ```
  CAST(NULL AS INTEGER)
  ```

パラメーター値を指定しない場合 (つまり、2 つの連続するコンマを指定する (**...,,...**))、このパラメーターは、ジオコーディングがセットアップされる時に指定するか、または該当のストアード・プロシージャーの *parameter_values* パラメーターを使用してバッチ・モードでジオコーディングを実行するときに、指定する必要があります。

このパラメーターのデータ・タイプは VARCHAR(32K) です。

autogeocoding_columns
トリガーを作成する列名のリストを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL で、自動ジオコーディングが使用可能になっている場合、表内のいずれかの列が更新されると、トリガーがアクティブ化されます。

autogeocoding_columns パラメーターに値を指定する場合は、任意の順序で列名を指定し、列名をコンマで区切ります。列名は、ジオコーディングされた列が存在する表と同じ表に存在しなければなりません。

このパラメーター設定値は、後続の自動ジオコーディングにのみ適用されます。

このパラメーターのデータ・タイプは VARCHAR(32K) です。

where_clause
WHERE 節の本体を指定し、これにより、ジオコーディングされるレコードのセットに対する制約事項を定義します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、WHERE 節に制限は定義されません。

節は、ジオコーダーの操作対象の表またはビューの、任意の列を参照することができます。キーワード WHERE は指定しないでください。

このパラメーター設定値は、後続のバッチ・モード・ジオコーディングにのみ適用されます。

このパラメーターのデータ・タイプは VARCHAR(32K) です。

commit_scope
n レコードをジオコーディングするたびに COMMIT を実行することを指定します。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、すべてのレコードをジオコーディングした後で COMMIT が行われます。

このパラメーター設定値は、後続のバッチ・モード・ジオコーディングにのみ適用されます。

このパラメーターのデータ・タイプは INTEGER です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_setup_geocoding ストアード・プロシージャーを呼び出す方法を示しています。 この例は DB2 CALL コマンドを使用して、CUSTOMER という名前の表の LOCATION という名前のジオコーディングされた列のジオコーディング処理をセットアップします。この CALL コマンドは、*geocoder_name* パラメーター値として DB2SE_USA_GEOCODER を指定します。

```
call db2gse.ST_setup_geocoding(NULL, 'CUSTOMERS', 'LOCATION',
    'DB2SE_USA_GEOCODER','ADDRESS,CITY,STATE,ZIP,1,100,80,,,,"$HOME/sqllib/
     gse/refdata/ky.edg","$HOME/sqllib/samples/extenders/spatial/EDGESample.loc"',
     'ADDRESS,CITY,STATE,ZIP',NULL,10,?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_unregister_geocoder

このストアード・プロシージャーは、DB2SE_USA_GEOCODER ジオコーダー (DB2 Spatial Extender と一緒に配布されるもの) 以外のジオコーダーの登録を抹消するために使用します。

制約事項:
ジオコーダーがいずれかの列のジオコーディング・セットアップに指定されている場合は、そのジオコーダーの登録を抹消することはできません。

あるジオコーダーが何らかの列のジオコーディング・セットアップに指定されているかどうかを判別するには、DB2GSE.ST_GEOCODING および DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビューをチェックします。登録を抹消しようとするジオコーダーについての情報を見つけるには、DB2GSE.ST_GEOCODERS カタログ・ビューを参照してください。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、登録を抹消するジオコーダーを含むデータベースに対して、DBADM 権限を持つ必要があります。

構文

パラメーターの説明

geocoder_name
ジオコーダーを一意的に指定します。このパラメーターには NULL でない値を指定する必要があります。

geocoder_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

出力パラメーター

msg_code
ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示

すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text
ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_unregister_geocoder ストアード・プロシージャーを呼び出す方法を示しています。 この例は DB2 CALL コマンドを使用して、SAMPLEGC という名前のジオコーダーの登録を抹消します。

```
call db2gse.ST_unregister_geocoder('SAMPLEGC',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

ST_unregister_spatial_column

このストアード・プロシージャーは、空間列の登録を抹消するために使用します。

このストアード・プロシージャーは、次のようにして登録を抹消します。

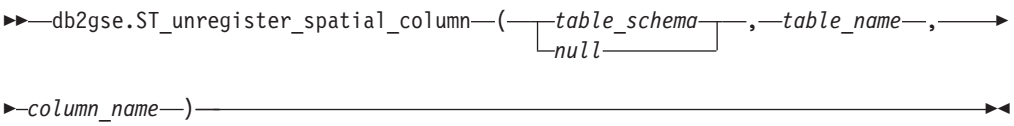
- 空間参照系と空間列の関連付けを除去する。ST_GEOMETRY_COLUMNS カタログ・ビューにはまだ空間列が含まれていますが、この列にはもう空間参照系は関連付けられていません。
- 基本表の場合、DB2 Spatial Extender がこの表に課した制約 (この空間列内の形状値が、すべて同じ空間参照系で表現されるように保証するための制約) をドロップする。

許可

このストアード・プロシージャーを呼び出すユーザー ID は、次の権限または特権の 1 つを持つ必要があります。

- DBADM 権限
- この表に対する CONTROL 特権
- この表に対する ALTER 特権

構文



パラメーターの説明

table_schema

table_name パラメーターに指定された表が属するスキーマの名前。このパラメーターには値を指定する必要がありますが、値は NULL にすることができます。このパラメーターが NULL の場合、表またはビューのスキーマ名として、CURRENT SCHEMA 特殊レジスター内の値が使用されます。

table_schema 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

table_name

column_name パラメーターに指定された列を含む表の、修飾されていない名前。このパラメーターには NULL でない値を指定する必要があります。

table_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

column_name

登録を抹消する空間列の名前。このパラメーターには NULL でない値を指定する必要があります。

column_name 値は、二重引用符で囲んだ場合を除き、英大文字に変換されます。

このパラメーターのデータ・タイプは VARCHAR(128) または、値を二重引用符で囲んだ場合は VARCHAR(130) です。

出力パラメーター

msg_code

ストアード・プロシージャーから戻されるメッセージ・コードを指定します。この出力パラメーターの値は、プロシージャーの処理中に起こる、エラー、成功、または警告状態を示します。このパラメーターの値が成功または警告の状態を示すものであれば、プロシージャーはそのタスクを終了します。パラメーター値がエラー状態を示すものであれば、データベースの変更は行われていません。

この出力パラメーターのデータ・タイプは INTEGER です。

msg_text

ストアード・プロシージャーから戻される、メッセージ・コードに付随する実際のメッセージ・テキストを指定します。メッセージ・テキストには、成功、警告、またはエラー状態についての追加情報 (エラーが起こった場所など) が含まれます。

この出力パラメーターのデータ・タイプは VARCHAR(1024) です。

例

この例は、DB2 コマンド行プロセッサーを使用して、ST_unregister_spatial_column ストアード・プロシージャーを呼び出す方法を示しています。 この例は、DB2 CALL コマンドを使用して、CUSTOMERS という名前の表の LOCATION という名前の空間列の登録を抹消しています。

```
call db2gse.ST_unregister_spatial_column(NULL,'CUSTOMERS','LOCATION',?,?)
```

この CALL コマンドの終わりの 2 つの疑問符は、出力パラメーター *msg_code* および *msg_text* を表します。これらの出力パラメーターの値は、ストアード・プロシージャーの実行後に表示されます。

第 21 章 カタログ・ビュー

Spatial Extender および Geodetic Data Management Feature のカタログ・ビューは有益な情報を提供します。

Spatial Extender のカタログ・ビューには、次の情報が含まれています。

251 ページの『DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビュー』
使用する座標システム

『DB2GSE.ST_GEOMETRY_COLUMNS カタログ・ビュー』
データを入れる、または更新する空間列

254 ページの『DB2GSE.ST_GEOCODERS カタログ・ビュー』 および 256 ページの『DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビュー』
使用するジオコーダー

255 ページの『DB2GSE.ST_GEOCODING カタログ・ビュー』 および 256 ページの『DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビュー』
ジオコーダーを自動的に実行するためのセットアップの指定および、バッチ・ジオコーディング中に実行する操作を前もって設定するための指定

258 ページの『DB2GSE.ST_SIZINGS カタログ・ビュー』
変数に割り当て可能な、値の最大長

258 ページの『DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビュー』 使用する空間参照系

260 ページの『DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビュー』
空間処理関数が生成した距離を表示する単位 (メートル、マイル、フィート、など)

DB2GSE.ST_GEOMETRY_COLUMNS カタログ・ビュー

データベースで、空間データを含むすべての表にある、すべての空間列に関する情報を調べるには、DB2GSE.ST_GEOMETRY_COLUMNS カタログ・ビューを使用します。

空間列が、空間参照系に関連付けられて登録されている場合は、このビューを使用して、空間参照系の名前と数値 ID を調べることもできます。空間列の追加情報については、DB2 の SYSCAT.COLUMN カタログ・ビューを照会します。

DB2GSE.ST_GEOMETRY_COLUMNS の説明については、次の表を参照してください。

表 29. DB2GSE.ST_GEOMETRY_COLUMNS カタログ・ビューの列

名前	データ・タイプ	NULL 可能かどうか	説明
TABLE_SCHEMA	VARCHAR(128)	いいえ	この空間列が入った表が属するスキーマの名前。
TABLE_NAME	VARCHAR(128)	いいえ	この空間列が入った表の非修飾名。

© Copyright IBM Corp. 1998, 2009

表 29. DB2GSE.ST_GEOMETRY_COLUMNS カタログ・ビューの列 (続き)

名前	データ・タイプ	NULL 可能かどうか	説明
COLUMN_NAME	VARCHAR(128)	いいえ	この空間列の名前。 TABLE_SCHEMA、TABLE_NAME、COLUMN_NAME の組み合わせで、列が一意的に識別される。
TYPE_SCHEMA	VARCHAR(128)	いいえ	この空間列の宣言されたデータ・タイプが属するスキーマの名前。この名前は DB2 カタログから得られる。
TYPE_NAME	VARCHAR(128)	いいえ	この空間列の宣言されたデータ・タイプの非修飾名。この名前は DB2 カタログから得られる。
SRS_NAME	VARCHAR(128)	はい	この空間列に関連した空間参照系の名前。その列に関連付けられた空間参照系がない場合、SRS_NAME は NULL である。
SRS_ID	INTEGER	はい	この空間列に関連した空間参照系の数値 ID。その列に関連付けられた空間参照系がない場合、SRS_ID は NULL である。

DB2GSE.SPATIAL_REF_SYS カタログ・ビュー

空間参照系を作成すると、DB2 Spatial Extender は、その ID および関係する情報をカタログ表に記録することにより登録します。この表から選択された列で DB2GSE.SPATIAL_REF_SYS カタログ・ビューが構成され、それが以下の表に説明されています。

表 30. DB2GSE.SPATIAL_REF_SYS カタログ・ビューの列

名前	データ・タイプ	NULL 可能かどうか	説明
SRID	INTEGER	いいえ	この空間参照系のユーザー定義 ID。
SR_NAME	VARCHAR(64)	いいえ	この空間参照系の名前。
CSID	INTEGER	いいえ	この空間参照系の基礎にある座標系の数値 ID。
CS_NAME	VARCHAR(64)	いいえ	この空間参照系の基礎にある座標系の名前。
AUTH_NAME	VARCHAR(256)	はい	この空間参照系の標準を設定した組織の名前。
AUTH_SRID	INTEGER	はい	AUTH_NAME 列に指定された組織が、この空間参照系に割り当てた ID。
SRTEXT	VARCHAR(2048)	いいえ	この空間参照系のアノテーション・テキスト。
FALSEX	FLOAT	いいえ	負の X 座標値から減算すると、負でない数 (つまり、正の数またはゼロ) になる数。
FALSEY	FLOAT	いいえ	負の Y 座標値から減算すると、負でない数 (つまり、正の数またはゼロ) になる数。
XYUNITS	FLOAT	いいえ	小数の X 座標または Y 座標に乗算すると、32 ビットのデータ項目として保管できる整数になる数。
FALSEZ	FLOAT	いいえ	負の Z 座標値から減算すると、負でない数 (つまり、正の数またはゼロ) になる数。
ZUNITS	FLOAT	いいえ	小数の Z 座標に乗算すると、32 ビットのデータ項目として保管できる整数になる数。
FALSEM	FLOAT	いいえ	負の指標から減算すると、負でない数 (つまり、正の数またはゼロ) になる数。

表 30. DB2GSE.SPATIAL_REF_SYS カタログ・ビューの列 (続き)

名前	データ・タイプ	NULL 可能かどうか	説明
MUNITS	FLOAT	いいえ	小数の指標に乗算すると、32 ビットのデータ項目として保管できる整数になる数。

DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビュー

登録済みの座標システムに関する情報を検索するには、DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューを参照してください。

Spatial Extender は、以下のタイミングで座標システムを Spatial Extender カタログに自動的に登録します。

- 空間操作にデータベースを使用できるようにすると、自動的に DB2 Spatial Extender カタログ表に登録されます。
- ユーザーが、データベースに追加の座標システムを定義します。

このビューの列の説明については、次の表を参照してください。

表 31. DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューの列

名前	データ・タイプ	NULL 可能かどうか	説明
COORDSYS_NAME	VARCHAR(128)	いいえ	この座標システムの名前。名前はデータベース内でユニークなもの。
COORDSYS_TYPE	VARCHAR(128)	いいえ	この座標システムのタイプ **PROJECTED** 2 ディメンション。 **GEOGRAPHIC** 3 ディメンション。X 座標と Y 座標を使用。 **GEOCENTRIC** 3 ディメンション。X、Y、Z の座標を使用。 **UNSPECIFIED** 抽象または非現実世界の座標システム。 この列の値は DEFINITION 列から得られる。
DEFINITION	VARCHAR(2048)	いいえ	この座標システムの定義の既知テキスト表記
ORGANIZATION	VARCHAR(128)	はい	この座標システムを定義した組織 (European Petrol Survey Group または ESPG のような標準団体) の名前。 ORGANIZATION_COORDSYS_ID 列が NULL の場合、この列は NULL。

表 31. DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューの列 (続き)

名前	データ・タイプ	NULL 可能かどうか	説明
ORGANIZATION_COORDSYS_ID	INTEGER	はい	座標システムを定義した組織によってこの座標システムに割り当てられた数値 ID。この ID と ORGANIZATION 列の値が両方とも NULL でない限り、この ID とこの列の値で座標システムが識別される。 ORGANIZATION 列が NULL の場合、ORGANIZATION_COORDSYS_ID 列も NULL である。
説明	VARCHAR(256)	はい	アプリケーションを示す座標システムの説明

DB2GSE.ST_GEOMETRY_COLUMNS カタログ・ビュー

データベースで、空間データを含むすべての表にある、すべての空間列に関する情報を調べるには、DB2GSE.ST_GEOMETRY_COLUMNS カタログ・ビューを使用します。

空間列が、空間参照系に関連付けられて登録されている場合は、このビューを使用して、空間参照系の名前と数値 ID を調べることもできます。空間列の追加情報については、DB2 の SYSCAT.COLUMN カタログ・ビューを照会します。

DB2GSE.ST_GEOMETRY_COLUMNS の説明については、次の表を参照してください。

表 32. DB2GSE.ST_GEOMETRY_COLUMNS カタログ・ビューの列

名前	データ・タイプ	NULL 可能かどうか	説明
TABLE_SCHEMA	VARCHAR(128)	いいえ	この空間列が入った表が属するスキーマの名前。
TABLE_NAME	VARCHAR(128)	いいえ	この空間列が入った表の非修飾名。
COLUMN_NAME	VARCHAR(128)	いいえ	この空間列の名前。 TABLE_SCHEMA、TABLE_NAME、COLUMN_NAME の組み合わせで、列が一意的に識別される。
TYPE_SCHEMA	VARCHAR(128)	いいえ	この空間列の宣言されたデータ・タイプが属するスキーマの名前。この名前は DB2 カタログから得られる。
TYPE_NAME	VARCHAR(128)	いいえ	この空間列の宣言されたデータ・タイプの非修飾名。この名前は DB2 カタログから得られる。
SRS_NAME	VARCHAR(128)	はい	この空間列に関連した空間参照系の名前。その列に関連付けられた空間参照系がない場合、SRS_NAME は NULL である。
SRS_ID	INTEGER	はい	この空間列に関連した空間参照系の数値 ID。その列に関連付けられた空間参照系がない場合、SRS_ID は NULL である。

DB2GSE.ST_GEOCODER_PARAMETERS カタログ・ビュー

データベースで空間操作を行えるようにすると、提供ジオコーダーの DB2GSE_USA_GEOCODER のパラメーターに関する情報が、自動的に DB2 Spatial Extender カタログに登録されます。追加のジオコーダーを登録すると、それらのパラメーターに関する情報もカタログに記録されます。カタログからジオコーダーのパラメーターに関する情報を検索するには、DB2GSE.ST_GEOCODER_PARAMETERS カタログ・ビューを参照してください。このビューの列の説明については、次の表を参照してください。

ジオコーダーのパラメーターについては、DB2 の SYSCAT.ROUTINEPARMS カタログ・ビューを参照してください。このビューの説明については、「SQL リファレンス」を参照してください。

表 33. DB2GSE.ST_GEOCODER_PARAMETERS の列

名前	データ・タイプ	NULL 可能かどうか	説明
GEOCODER_NAME	VARCHAR(128)	いいえ	このパラメーターをもつジオコーダーの名前。
ORDINAL	SMALLINT	いいえ	COLUMN_NAME 列で指定されるジオコーダーの働きをする関数のシグニチャーでの、このパラメーター (つまり、PARAMETER_NAME 列で指定されたパラメーター) の位置。 GEOCODER_NAME 列と ORDINAL 列の結合値によって、このパラメーターは一意的に識別される。 DB2 の SYSCAT.ROUTINEPARMS カタログ・ビューのレコードにも、このパラメーターに関する情報がある。このレコードには、SYSCAT.ROUTINEPARMS の ORDINAL 列に表示される値が含まれている。この値は、DB2GSE.ST_GEOCODER_PARAMETERS ビューの ORDINAL 列に表示されるものと同じである。
PARAMETER_NAME	VARCHAR(128)	はい	このパラメーターの名前。このパラメーターをもつ関数が作成されたときに、名前が指定されなかった場合は、PARAMETER_NAME 列は NULL である。 PARAMETER_NAME 列の内容は、DB2 カタログから得られる。
TYPE_SCHEMA	VARCHAR(128)	いいえ	このパラメーターをもつスキーマの名前。この名前は DB2 カタログから得られる。
TYPE_NAME	VARCHAR(128)	いいえ	このパラメーターに割り当てられた値のデータ・タイプの非修飾名。この名前は DB2 カタログから得られる。

表 33. DB2GSE.ST_GEOCODER_PARAMETERS の列 (続き)

名前	データ・タイプ	NULL 可能かどうか	説明
PARAMETER_DEFAULT	VARCHAR(2048)	はい	このパラメーターに割り当てられるデフォルト値。DB2 はこの値を SQL 式として解釈する。 値が引用符で囲まれている場合は、ジオコーダーにストリングとして渡される。そうでない場合は、パラメーターがジオコーダーに渡されるときに、SQL 式の計算によって、パラメーターのデータ・タイプが決められる。PARAMETER_DEFAULT 列に NULL が入っていると、この NULL 値がジオコーダーに渡される。 デフォルト値は、DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビューで対応する値をもつ場合がある。また、ST_run_geocoding ストアード・プロシージャーに対する入力でも、対応する値をもつ場合がある。対応するいずれかの値が、デフォルト値と異なる場合は、対応する値がデフォルト値をオーバーライドする。
DESCRIPTION	VARCHAR(256)	はい	アプリケーションを示すパラメーターの説明。

DB2GSE.ST_GEOCODERS カタログ・ビュー

空間操作にデータベースを使用できるようにすると、提供ジオコーダーの DB2GSE_USA_GEOCODER が、自動的に DB2 Spatial Extender カタログ表に登録されます。 ユーザーに対し、追加のジオコーダーを使用可能にしたい場合は、これらのジオコーダーを登録する必要があります。登録済みのジオコーダーに関する情報を検索するには、DB2GSE.ST_GEOCODERS カタログ・ビューを照会してください。このビューの列の説明については、次の表を参照してください。

ジオコーダーのパラメーターについては、DB2 Spatial Extender の DB2GSE.ST_GEOCODER_PARAMETERS カタログ・ビューと DB2 の SYSCAT.ROUTINEPARMS カタログ・ビューを照会してください。 ジオコーダーとして使用される関数については、DB2 の SYSCAT.ROUTINES カタログ・ビューを照会してください。

表 34. DB2GSE.ST_GEOCODERS カタログ・ビューの列

名前	データ・タイプ	NULL 可能かどうか	説明
GEOCODER_NAME	VARCHAR(128)	いいえ	このジオコーダーの名前。データベース内でユニークな名前。
FUNCTION_SCHEMA	VARCHAR(128)	いいえ	このジオコーダーとして使用されている関数が属するスキーマの名前。
FUNCTION_NAME	VARCHAR(128)	いいえ	このジオコーダーとして使用されている関数の非修飾名。

表 34. DB2GSE.ST_GEOCODERS カタログ・ビューの列 (続き)

名前	データ・タイプ	NULL 可能かどうか	説明
SPECIFIC_NAME	VARCHAR(128)	いいえ	このジオコーダーとして使用されている関数の特定名。 FUNCTION_SCHEMA と SPECIFIC_NAME の結合値によって、このジオコーダーとして使用されている関数が一意的に識別される。
RETURN_TYPE_SCHEMA	VARCHAR(128)	いいえ	このジオコーダーの出力パラメーターのデータ・タイプが属するスキーマの名前。この名前は DB2 カタログから得られる。
RETURN_TYPE_NAME	VARCHAR(128)	いいえ	このジオコーダーの出力パラメーターのデータ・タイプの非修飾名。この名前は DB2 カタログから得られる。
VENDOR	VARCHAR(256)	はい	このジオコーダーを作成したベンダーの名前。
説明	VARCHAR(256)	はい	アプリケーションを示すジオコーダーの説明

DB2GSE.ST_GEOCODING カタログ・ビュー

ジオコーディング操作をセットアップすると、個々の設定は、自動的に DB2 Spatial Extender カタログに記録されます。これらの設定項目を調べるには、DB2GSE.ST_GEOCODING と DB2GSE.ST_GEOCODING_PARAMETERS のカタログ・ビューを照会してください。次の表で説明する DB2GSE.ST_GEOCODING カタログ・ビューには、すべての設定項目が含まれています。例えば、各コミット前にジオコーダーが処理するレコード数などです。

DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビューには、各ジオコーダーに固有の項目が含まれています。例えば、提供ジオコーダーの DB2GSE_USA_GEOCODER の設定項目には、ジオコーダーが入力をジオコーディングする際に、入力として与えられたアドレスと実際のアドレスが一致しなければならない最小度合いが含まれています。最小一致スコアと呼ばれるこの最小必要要件は、DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビューに記録されています。

表 35. DB2GSE.ST_GEOCODING カタログ・ビューの列

名前	データ・タイプ	NULL 可能かどうか	説明
TABLE_SCHEMA	VARCHAR(128)	いいえ	COLUMN_NAME 列で識別される列を含む表が入ったスキーマの名前。
TABLE_NAME	VARCHAR(128)	いいえ	COLUMN_NAME 列で識別される列を含む表の非修飾名。
COLUMN_NAME	VARCHAR(128)	いいえ	このカタログ・ビューに示される指定に従ってデータが読み込まれる空間列の名前。 TABLE_SCHEMA、TABLE_NAME、COLUMN_NAME の列の結合値によって、空間列が一意的に識別される。

表 35. DB2GSE.ST_GEOCODING カタログ・ビューの列 (続き)

名前	データ・タイプ	NULL 可能かどうか	説明
GEOCODER_NAME	VARCHAR(128)	いいえ	COLUMN_NAME 列で指定される空間列にデータを生成するジオコーダーの名前。 1 つの空間列に割り当てることができるのは、1 つのジオコーダーだけである。
MODE	VARCHAR(128)	いいえ	ジオコーディング処理のモード: **BATCH** バッチ・ジオコーディングだけが可能。 **AUTO** 自動ジオコーディングがセットアップされており、アクティブ化されている。 **INVALID** 空間カタログ表で矛盾を検出。ジオコーディング入力は無効。
SOURCE_COLUMNS	VARCHAR(10000)	はい	自動ジオコーディング用にセットアップされた表列の名前。これらの列が更新されると、トリガーが、ジオコーダーに更新データをジオコーディングするように常に促す。
WHERE_CLAUSE	VARCHAR(10000)	はい	WHERE 節内の検索条件。この条件は、ジオコーダーがバッチ・モードで実行される場合、ジオコーダーが、レコードの指定されたサブセット内のデータだけをジオコーディングするように指示する。
COMMIT_COUNT	INTEGER	はい	コミットが出される前にバッチ・ジオコーディングで処理される行数。 COMMIT_COUNT 列の値が 0 (ゼロ) または NULL の場合、コミットは出されない。

DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビュー

特定のジオコーダーにジオコーディング操作をセットアップすると、ジオコーダーに固有の設定は、自動的に Spatial Extender カタログに記録されます。例えば、提供ジオコーダー DB2GSE_USA_GEOCODER に固有の操作は、入力として与えられたアドレスを参照データと比較し、指定された度合い、またはそれより高い度合いでそれらが一致する場合は、前者のアドレスをジオコーディングすることです。このジオコーダーに操作をセットアップする場合は、最小一致スコアと呼ばれる度合いを指定します。指定された値はカタログに記録されます。

ジオコーディング操作についてのジオコーダーに固有の設定を調べるには、DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビューを照会してください。このビューは、次の表で説明されています。

ジオコーディング操作のセットアップについての特定のデフォルトは、DB2GSE.ST_GEOCODER_PARAMETERS カタログ・ビューで確認することができます。 DB2GSE.ST_GEOCODING_PARAMETERS ビューの値はデフォルトをオーバーライドします。

表 36. DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビューの列

名前	データ・タイプ	NULL 可能かどうか	説明
TABLE_SCHEMA	VARCHAR(128)	いいえ	COLUMN_NAME 列で識別される列を含む表が入ったスキーマの名前。
TABLE_NAME	VARCHAR(128)	いいえ	空間列が入った表の非修飾名。
COLUMN_NAME	VARCHAR(128)	いいえ	このカタログ・ビューに示される指定に従ってデータが読み込まれる空間列の名前。 TABLE_SCHEMA、TABLE_NAME、COLUMN_NAME の列の結合値によって、空間列が一意的に識別される。
ORDINAL	SMALLINT	いいえ	COLUMN_NAME 列で識別される列に対してジオコーダーとして動作する関数のシグニチャーの中でのこのパラメーター (つまり、PARAMETER_NAME 列で指定されたパラメーター) の位置。 DB2 の SYSCAT.ROUTINEPARMS カタログ・ビューのレコードにも、このパラメーターに関する情報がある。このレコードには、SYSCAT.ROUTINEPARMS の ORDINAL 列に表示される値が含まれている。この値は、DB2GSE.ST_GEOCODING_PARAMETERS ビューの ORDINAL 列に表示されるものと同じである。
PARAMETER_NAME	VARCHAR(128)	はい	ジオコーダーの定義におけるパラメーター名。ジオコーダーが定義されたときに名前が指定されなかった場合は、PARAMETER_NAME は NULL である。 PARAMETER_NAME 列のこの内容は、DB2 カタログから得られる。
PARAMETER_VALUE	VARCHAR(2048)	はい	このパラメーターに割り当てられる値。DB2 はこの値を SQL 式として解釈する。 値が引用符で囲まれている場合は、ジオコーダーにストリングとして渡される。そうでない場合は、パラメーターがジオコーダーに渡されるときに、SQL 式の評価によって、パラメーターのデータ・タイプが決められる。PARAMETER_VALUE 列に NULL が入っていると、この NULL がジオコーダーに渡される。 PARAMETER_VALUE 列は、DB2GSE.ST_GEOCODER_PARAMETERS カタログ・ビューの PARAMETER_DEFAULT 列に対応する。PARAMETER_VALUE 列に値が入る場合は、この値は、PARAMETER_DEFAULT 列のデフォルト値をオーバーライドする。 PARAMETER_VALUE 列が NULL の場合は、デフォルト値が使用される。

DB2GSE.ST_SIZINGS カタログ・ビュー

以下のものを検索するには、DB2GSE.ST_SIZINGS カタログ・ビューを使用します。

- Spatial Extender がサポートするすべての変数。例えば、座標システムの名前、ジオコーダーの名前、および空間データの事前割り当てテキスト表記を割り当てることのできる変数などです。
- あらかじめわかっている場合、これらの変数に割り当てられた値の許容最大長 (例えば、座標システム名、ジオコーダー名、空間データの事前割り当てテキスト表記の許容最大長など)。

ビューの列の説明については、次の表を参照してください。

表 37. DB2GSE.ST_SIZINGS カタログ・ビューの列

名前	データ・タイプ	NULL 可能かどうか	説明
VARIABLE_NAME	VARCHAR(128)	いいえ	変数を示す用語。用語はデータベース内でユニークなもの。
SUPPORTED_VALUE	INTEGER	はい	VARIABLE_NAME 列に示された変数に割り当てられた値の許容最大長。SUPPORTED_VALUE 列に使用できる値は、以下のとおりです。 **ゼロ以外の数値** この変数に割り当てられた値の許容最大長。 **0** どの長さも許容される。あるいは、許容長を判別できない。 **NULL** Spatial Extender では、この変数はサポートされていない。
DESCRIPTION	VARCHAR(128)	はい	この変数の説明。

DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビュー

登録済みの空間参照系に関する情報を検索するには、DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビューを照会してください。Spatial Extender は、以下のタイミングで空間参照系を Spatial Extender カタログに自動的に登録します。

- ユーザーが、データベースを空間操作に使用できるようにし、5 つのデフォルト空間参照系と 318 の定義済み測地参照系を使用可能にした時。
- ユーザーが追加の空間参照系を作成した時。

DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビューからすべての値を入手するには、それぞれの空間参照系が座標システムに関連付けられていることを理解しておく必要があります。空間参照系は、座標システムから導出された座標を DB2 が最大の効率で処理できる値に変換するように、またこれらの座標が参照できるスペースの最大可能範囲を定義するように設計されています。

特定の空間参照系に関連付けられた座標システムの名前とタイプを確認するには、DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビューの COORDSYS_NAME 列と COORDSYS_TYPE 列を照会してください。座標システムの詳細については、DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューを照会してください。

表 38. DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビューの列

名前	データ・タイプ	NULL 可能かどうか	説明
SRS_NAME	VARCHAR(128)	いいえ	空間参照系の名前。名前はデータベース内でユニークなもの。
SRS_ID	INTEGER	いいえ	空間参照系の数値 ID。各空間参照系はユニークな数値 ID をもっている。測地参照系の SRS_ID 値の範囲は 2000000000 から 2000001000 までの間。 空間処理関数は、名前ではなく、数値 ID によって空間参照系を指定する。
X_OFFSET	DOUBLE	いいえ	形状のすべての X 座標から減算されるオフセット。減算は、形状座標を、DB2 が最大の効率で処理できる値に変換するプロセスの 1 ステップである。次のステップで、減算で得られた数値に、X_SCALE 列に示されたスケール係数が乗算される。
X_SCALE	DOUBLE	いいえ	X 座標からオフセットを減算して得られた数値に乗算されるスケール係数。この係数は Y_SCALE 列に示された値と同一である。
Y_OFFSET	DOUBLE	いいえ	形状のすべての Y 座標から減算されるオフセット。減算は、形状座標を、DB2 が最大の効率で処理できる値に変換するプロセスの 1 ステップである。次のステップで、減算で得られた数値に、Y_SCALE 列に示されたスケール係数が乗算される。
Y_SCALE	DOUBLE	いいえ	Y 座標からオフセットを減算して得られた数値に乗算されるスケール係数。この係数は X_SCALE 列に示された値と同一である。
Z_OFFSET	DOUBLE	いいえ	形状のすべての Z 座標から減算されるオフセット。減算は、形状座標を、DB2 が最大の効率で処理できる値に変換するプロセスの 1 ステップである。次のステップで、減算で得られた数値に、Z_SCALE 列に示されたスケール係数が乗算される。
Z_SCALE	DOUBLE	いいえ	Z 座標からオフセットを減算して得られた数値に乗算されるスケール係数。
M_OFFSET	DOUBLE	いいえ	形状に関連するすべての指標から減算されるオフセット。減算は、指標を、DB2 が最大の効率で処理できる値に変換するプロセスの 1 ステップである。次のステップで、減算で得られた数値に、M_SCALE 列に示されたスケール係数が乗算される。
M_SCALE	DOUBLE	いいえ	指標からオフセットを減算して得られた数値に乗算されるスケール係数。
MIN_X	DOUBLE	いいえ	この空間参照系が適用される、形状の X 座標の可能最小値。この値は X_OFFSET 列と X_SCALE 列の値から導出される。

表 38. DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビューの列 (続き)

名前	データ・タイプ	NULL 可能かどうか	説明
MAX_X	DOUBLE	いいえ	この空間参照系が適用される、形状の X 座標の可能最大値。この値は X_OFFSET 列と X_SCALE 列の値から導出される。
MIN_Y	DOUBLE	いいえ	この空間参照系が適用される、形状の Y 座標の可能最小値。この値は Y_OFFSET 列と Y_SCALE 列の値から導出される。
MAX_Y	DOUBLE	いいえ	この空間参照系が適用される、形状の Y 座標の可能最大値。この値は Y_OFFSET 列と Y_SCALE 列の値から導出される。
MIN_Z	DOUBLE	いいえ	この空間参照系が適用される、形状の Z 座標の可能最小値。この値は Z_OFFSET 列と Z_SCALE 列の値から導出される。
MAX_Z	DOUBLE	いいえ	この空間参照系が適用される、形状の Z 座標の可能最大値。この値は Z_OFFSET 列と Z_SCALE 列の値から導出される。
MIN_M	DOUBLE	いいえ	この空間参照系が適用される、形状と一緒に保管できる指標の可能最小値。この値は M_OFFSET 列と M_SCALE 列の値から導出される。
MAX_M	DOUBLE	いいえ	この空間参照系が適用される、形状と一緒に保管できる指標の可能最大値。この値は M_OFFSET 列と M_SCALE 列の値から導出される。
COORDSYS_NAME	VARCHAR(128)	いいえ	この空間参照系の基礎となる座標システムの識別名。
COORDSYS_TYPE	VARCHAR(128)	いいえ	この空間参照系の基礎となる座標システムのタイプ。
ORGANIZATION	VARCHAR(128)	はい	この空間参照系の基礎となる座標システムを定義した組織名 (標準化団体など)。 ORGANIZATION_COORSYS_ID が NULL の場合は ORGANIZATION は NULL である。
ORGANIZATION_ COORDSYS_ID	INTEGER	はい	この空間参照系の基礎となる座標システムを定義した組織名 (標準化団体など)。 ORGANIZATION が NULL の場合は ORGANIZATION_COORDSYS_ID は NULL である。
DEFINITION DESCRIPTION	VARCHAR(2048) VARCHAR(256)	いいえ はい	座標システムの定義の事前割り当てテキスト表記 空間参照系の説明。

DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビュー

特定の空間処理関数は、特定の距離を表す値を受け取ったり、戻したりします。いくつかの場合では、距離を表す測定単位を選択できます。例えば、ST_Distance は、2 つの指定した形状間の最小距離を戻します。この場合、ST_Distance に対して、マイルによる距離を要求することもできるし、メートルによる距離を要求することもできます。選択可能な測定単位を確認するには、DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビューを参照してください。

表 39. DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビューの列

名前	データ・タイプ	NULL 可能かどうか	説明
UNIT_NAME	VARCHAR(128)	いいえ	測定単位の名前。名前はデータベース内でユニークなもの。
UNIT_TYPE	VARCHAR(128)	いいえ	測定単位のタイプ。使用できる値は以下のとおりです。 **LINEAR** 線の測定単位。 **ANGULAR** 角度の測定単位。
CONVERSION_FACTOR	DOUBLE	いいえ	この測定単位を基本単位に変換する数値。線の測定単位の基本単位はメートルであり、角度の測定単位の基本単位はラジアンである。 基本単位自身の変換係数は 1.0。
DESCRIPTION	VARCHAR(256)	はい	測定単位の説明。

第 22 章 空間処理関数: カテゴリーおよび使用法

この章では、すべての空間処理関数をカテゴリー別に紹介しています。

空間処理関数: カテゴリーおよび使用法

DB2® Spatial Extender は、次のことを行う関数を提供します。

- 形状を様々なデータ交換フォーマットとの間で変換する。これらの関数は、コンストラクター関数 と呼ばれています。
- 境界、交差、その他の情報について形状を比較する。これらの関数は、比較関数と呼ばれています。
- 形状内の座標や指標、形状間の関係、および境界やその他の情報など、形状のプロパティーに関する情報を戻す。
- 既存の形状から新規の形状を生成する。
- 形状内の点と点の最短距離を測る。
- 索引パラメーターについての情報を提供する。
- 異なる座標システムの間で投影や変換を行う。

形状値をデータ交換フォーマットに変換する空間処理関数

DB2 Spatial Extender は、形状を以下のデータ交換フォーマットとの間で変換する空間処理関数を提供します。

- 事前割り当てテキスト (WKT) 表記
- 事前割り当てバイナリー (WKB) 表記
- ESRI 形状表記
- Geography Markup Language (GML) 表記

これらのフォーマットから形状を作成する関数は、*コンストラクター関数* として知られています。

コンストラクター関数

DB2 Spatial Extender は、形状を以下のデータ交換フォーマットとの間で変換する空間処理関数を提供します。

- 事前割り当てテキスト (WKT) 表記
- 事前割り当てバイナリー (WKB) 表記
- ESRI 形状表記
- Geography Markup Language (GML) 表記

これらのフォーマットから形状を作成する関数は、*コンストラクター関数* として知られています。コンストラクター関数は、データが挿入される列の形状データ・タ

© Copyright IBM Corp. 1998, 2009

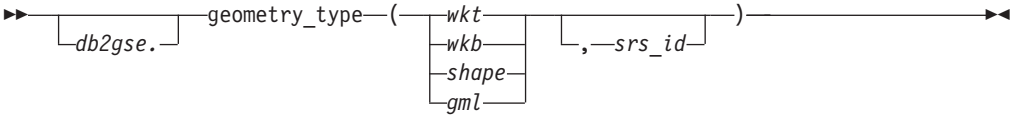
イプと同じ名前をもっています。これらの関数は、常にそれぞれの入力データ交換フォーマットに関して働きます。このセクションでは、以下について述べます。

- データ交換フォーマットに関して働く呼び出し関数の SQL、およびそれらの関数が戻す形状のタイプ
- X および Y 座標からポイントを作成する呼び出し関数の SQL、およびその関数が戻す形状のタイプ
- コードおよび結果セットの例

データ交換フォーマットで作動する関数

このセクションでは、データ交換フォーマットで作動する関数の構文を紹介し、関数の入力パラメーターを説明し、これらの関数が戻す形状のタイプを明らかにします。

構文



パラメーターおよびその他の構文エレメント

db2gse DB2® Spatial Extender によって提供される空間データ・タイプが属するスキーマの名前。

geometry_type
以下のコンストラクター関数の 1 つ。

- ST_Point
- ST_LineString
- \
- ST_Polygon
- ST_MultiPoint
- ST_MultiLineString
- ST_MultiPolygon
- ST_GeomCollection
- ST_Geometry

wkt 形状の事前割り当てテキスト表記が入る、CLOB(2G) タイプの値。

wkb 形状の事前割り当てバイナリー表記が入る、BLOB(2G) タイプの値。

shape 形状の ESRI 形状表記が入る、BLOB(2G) タイプの値。

gml 形状の Geography Markup Language (GML) 表記が入る、CLOB(2G) タイプの値。

srs_id 結果の形状の空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

戻りタイプ

geometry_type

geometry_type が *ST_Geometry* の場合、戻された形状タイプの動的タイプは、入力値によって示された形状に対応します。

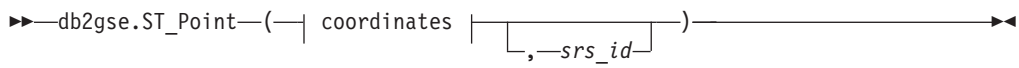
geometry_type がそれ以外のタイプの場合、戻された形状タイプの動的タイプは、関数名に対応します。入力値によって示された形状が、関数名またはそのサブタイプの 1 つの名前に合致しない場合、エラーが戻されます。

座標から形状を作成する関数

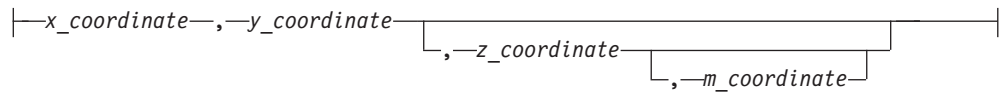
このセクションでは、ST_Point を呼び出す構文、そのパラメーターの説明、およびそれが戻す形状のタイプについての情報を記載しています。

ST_Point 関数は、データ交換フォーマットからだけでなく、数値座標値からも形状を作成します。これは、ロケーション・データがすでにユーザーのデータベースに保管されている場合は、非常に便利な機能です。

構文



座標:



パラメーター

x_coordinate
結果のポイントの X 座標を示す、DOUBLE タイプの値。

y_coordinate
結果のポイントの Y 座標を示す、DOUBLE タイプの値。

z_coordinate
結果のポイントの Z 座標を示す、DOUBLE タイプの値。

z_coordinate パラメーターを省略すると、結果のポイントは Z 座標を持ちません。このようなポイントの場合、ST_Is3D の結果は 0 (ゼロ) です。

m_coordinate
結果のポイントの M 座標を示す、DOUBLE タイプの値。

m_coordinate パラメーターを省略すると、結果のポイントは指標を持ちません。このようなポイントの場合 ST_IsMeasured の結果は 0 (ゼロ) です。

srs_id 結果のポイントの空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_Point

例

このセクションでは、コンストラクター関数を呼び出すコード、コンストラクター関数の出力を入れるための表を作成するコード、出力を検索するコード、および出力そのものの例を示します。

次の例は、事前割り当て (WKT) 表記を使用した座標表記を使用して、 ID 100 および X 座標 30、Y 座標 40 のポイント値を使用して、空間参照系 1 で、SAMPLE_GEOMETRY 表に行を挿入します。その次に、ID 200 および示された座標の折れ線値で、別の行を挿入します。

```
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"
```

空間列に入れることができるのは ST_Point 値だけであることが分かっている場合は、 2 つのポイントを挿入する以下のような例を使用することができます。折れ線、またはポイントではない別のタイプを挿入しようとすると、SQL エラーになります。最初の挿入は、事前割り当てテキスト表記 (WKT) からポイント形状を作成します。2 番目の挿入は、数値座標値からポイント形状を作成します。これらの入力値は、既存の表列からも選択できることに注意してください。

```
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"
```

次の例では、組み込み SQL が使用されていて、アプリケーションがデータ域を、該当する値で埋めることを想定しています。

```
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;

// * Application logic to read into buffers goes here */

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));
```

次のサンプル Java™ コードでは、 X、Y 数値座標値を使ってポイント形状を挿入するために JDBC を使用し、形状を指定するのに WKT 表記を使用しています。

```
String ins1 = "INSERT into sample_geometry (id, geom)
               VALUES(?, db2gse.ST_PointFromText(CAST( ?
               as VARCHAR(128)), 1))";
PreparedStatement pstmt = con.prepareStatement(ins1);
pstmt.setInt(1, 100);                     // id value
pstmt.setString(2, "point(32.4 50.7)");  // wkt value
int rc = pstmt.executeUpdate();

String ins2 = "INSERT into sample_geometry (id, geom)
               VALUES(?, db2gse.ST_Point(CAST( ? as double),
               CAST(? as double), 1))";
pstmt = con.prepareStatement(ins2);
pstmt.setInt(1, 200);         // id value
pstmt.setDouble(2, 40.3);     // lat
pstmt.setDouble(3, -72.5);    // long
rc = pstmt.executeUpdate();
```

事前割り当てテキスト (WKT) 表記への変換

テキスト表記は、ASCII 文字ストリングを表す CLOB 値です。この表記では、形状を ASCII テキスト・フォームで交換できます。

ST_AsText 関数は、表に保管されている形状値を WKT ストリングに変換します。次の例では、簡単なコマンド行照会を使用して、以前に SAMPLE_GEOMETRY 表に挿入された値を選択しています。

```
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)
```

次の例では、組み込み SQL を使用して、以前に SAMPLE_GEOMETRY 表に挿入された値を選択しています。

```
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;
```

代わりに、ST_WellKnownText トランスフォーム・グループを使用して、形状をバインドアウトするときに、それら形状を、その事前割り当てテキスト表記に暗黙的に変換することができます。次のコード例は、トランスフォーム・グループの使用方法について説明しています。

```
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;
```

SELECT ステートメントでは、形状を変換するために空間処理関数が使用されることはありません。

このセクションで説明した関数のほかに、DB2® Spatial Extender は、事前割り当てテキスト表記との間で形状を変換するその他の関数も提供しています。 DB2 Spatial Extender は、OGC 『Simple Features for SQL』 仕様および ISO SQL/MM Part 3: Spatial 標準をインプリメントするための、他の関数を提供しています。 これらの関数には以下のものがあります。

- **ST_WKTToSQL**
- **ST_GeomFromText**
- **ST_GeomCollFromTxt**
- **ST_PointFromText**
- **ST_LineFromText**
- **ST_PolyFromText**
- **ST_MPointFromText**
- **ST_MLineFromText**
- **ST_MPolyFromText**

事前割り当てバイナリー (WKB) 表記への変換

WKB 表記は、BLOB 値であるバイナリー・データ構造で構成されています。これらの BLOB 値はバイナリー・データ構造を表しており、この構造は、DB2® がサポートし、かつ DB2 が言語バインディングを持つプログラム言語で書かれたアプリケーション・プログラムが管理する必要があります。

ST_AsBinary 関数は、表に保管されている形状値を、プログラム・ストレージ内の BLOB 変数にフェッチできる、事前割り当てバイナリー (WKB) 表記に変換します。次の例では、組み込み SQL を使用して、以前に SAMPLE_GEOMETRY 表に挿入された値を選択しています。

```
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;
```

代わりに、ST_WellKnownBinary トランスフォーム・グループを使用して、形状をバインドアウトするときに、それら形状をその事前割り当てバイナリー表記に暗黙的に変換することができます。次のコード例は、このトランスフォーム・グループの使用方法について説明しています。

```
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;
```

SELECT ステートメントでは、形状を変換するために空間処理関数が使用されることはありません。

このセクションで説明した関数のほかに、事前割り当てバイナリー表記との間で形状を変換するその他の関数もあります。 DB2 Spatial Extender は、OGC 『Simple Features for SQL』 仕様および ISO SQL/MM Part 3: Spatial 標準をインプリメントするための、他の関数を提供しています。 これらの関数には以下のものがあります。

- **ST_WKBToSQL**
- **ST_GeomFromWKB**
- **ST_GeomCollFromWKB**
- **ST_PointFromWKB**

- **ST_LineFromWKB**
- **ST_PolyFromWKB**
- **ST_MPointFromWKB**
- **ST_MLineFromWKB**
- **ST_MPolyFromWKB**

ESRI 形状表記への変換

ESRI 形状表記は、サポートされている言語で書かれた、アプリケーション・プログラムによって管理する必要のある、バイナリー・データ構造で構成されています。

ST_AsShape 関数は、表に保管されている形状値を、プログラム・ストレージの BLOB 変数にフェッチできる、 ESRI 形状表記に変換します。次の例では、組み込み SQL を使用して、以前に SAMPLE_GEOMETRY 表に挿入された値を選択しています。

```
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;
```

代わりに、ST_Shape トランスフォーム・グループを使用して、形状をバインドアウトするときに、それら形状を、その形状表記に暗黙的に変換することができます。次のコード例は、トランスフォーム・グループの使用方法について説明しています。

```
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;
```

SELECT ステートメントでは、形状を変換するために空間処理関数が使用されることはありません。

Geography Markup Language (GML) 表記への変換

Geography Markup Language (GML) 表記は ASCII ストリングです。この表記では、形状を ASCII テキスト・フォームで交換できます。

ST_AsGML 関数は、表に保管されている形状値を GML テキスト・ストリングに変換します。次の例では、以前に SAMPLE_GEOMETRY 表に挿入された値を選択

しています。例に示されている結果は、読みやすいようにフォーマットし直されています。 結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

```
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>
```

代わりに、ST_GML トランスフォーム・グループを使用して、形状をバインドアウトするときに、それら形状を、その HTML 表記に暗黙的に変換することができます。

```
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>
```

SELECT ステートメントでは、形状を変換するために空間処理関数が使用されることはありません。

地形を比較する関数

一部の空間処理関数は、地勢が互いに関係し合ったり、比較し合ったりする方法に関する情報を戻します。他の空間処理関数は、座標システムの 2 つの定義または 2 つの空間参照系が同一であるかどうかについての情報を戻します。すべての場合において、戻される情報は、形状間、座標システムの定義間、または空間参照系間での比較の結果です。この情報を提供する関数は、比較関数と呼ばれています。

表 40. 目的別の比較関数

目的	関数
一方の形状の内部が他方の内部と交差するかどうかを判別します。	• ST_Contains • ST_Within
形状の交差についての情報を戻します。	• ST_Crosses • ST_Intersects • ST_Overlaps • ST_Touches

表 40. 目的別の比較関数 (続き)

目的	関数
1 つの形状を囲む最小外接長方形が、もう 1 つの形状を囲む最小外接長方形と交差するかどうかを判別します。	• ST_EnvIntersects • ST_MBRIntersects
2 つのオブジェクトが同一であるかどうかを判別します。	• ST_Equals • ST_EqualCoordsys • ST_EqualSRS
比較中の形状が DE-9IM パターン・マトリックス・ストリングの条件と一致するかどうかを判別します。	• ST_Relate
2 つの形状が交差しているかどうかをチェックします。	• ST_Disjoint

比較関数

DB2® Spatial Extender の比較関数は、比較が一定の基準に合致した場合は値 1、比較がその基準に合致しなかった場合は値 0 (ゼロ)、および比較が実行できなかった場合は NULL 値を戻します。入力パラメーターに対して比較操作が定義されていない場合、またはいずれかのパラメーターが NULL である場合、比較は実行できません。異なるデータ・タイプまたはディメンションをもつ形状がパラメーターに割り当てられていても、比較は実行できます。

Dimensionally Extended 9 Intersection Model (DE-9IM) は、異なるタイプとディメンションの形状間で、ペアワイズ (2 つ一組の) 空間リレーションシップを定義する数学的アプローチです。このモデルは、すべてのタイプの形状間の空間のリレーションシップを、結果として生じる交差のディメンションを考慮に入れて、それらの内部、境界および外部のペアワイズ交差として表しています。

形状 *a* と *b* があると仮定します。I(*a*)、B(*a*)、および E(*a*) が、それぞれ *a* の内部、境界、および外部を表しています。また、I(b)、B(b)、および E(b) が、*b* の内部、境界、および外部を表しています。I(a)、B(a)、および E(a) と I(b)、B(b)、および E(b) との交差は、3 x 3 のマトリックスになります。それぞれの交差は、異なるディメンションの形状になります。例えば、2 つのポリゴンの境界の交差は、ポイントと折れ線から構成されます。この場合、dim 関数は最大ディメンション 1 を戻します。

dim 関数は、–1、0、1 または 2 の値を戻します。–1 は NULL 集合または dim(null) に相当します。これは交差が検出されない場合に戻されます。

比較関数によって戻された結果は、DE-9IM の許容値を表すパターン・マトリックスと、比較関数によって戻された結果を比較することによって、理解し、検証することができます。

パターン・マトリックスには、各交差マトリックス・セルの許容値が含まれています。可能なパターン値は、以下のとおりです。

T 交差がなければならない。dim = 0、1、または 2。

F 交差があってはならない。dim = -1。

***** 交差があっても構わない。dim = -1、0、1、または 2。

0 交差がなければならず、そのディメンションは 0 でなければならない。dim = 0。

1 交差がなければならず、その最大ディメンションは 1 でなければならない。dim = 1。

2 交差がなければならず、その最大ディメンションは 2 でなければならない。dim = 2。

例えば、以下の ST_Within 関数のパターン・マトリックスには値 T、F、および * が含まれています。

表 41. ST_Within のマトリックス： 形状組み合わせについての ST_Within 関数のパターン・マトリックス

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 内部	T	*	F
形状 a 境界	*	*	F
形状 a 外部	*	*	*

ST_Within 関数は、両方の形状の内部が交差する場合、および *a* の内部または境界が *b* の外部と交差しない場合、値 1 を戻します。他のすべての条件は関係ありません。

各関数は少なくとも 1 つのパターン・マトリックスをもっています。しかし、中には色々な形状タイプの組み合わせのリレーションシップを記述するために、複数のパターン・マトリックスを必要とするものがあります。

DE-9IM は Clementini および Felice によって開発されました。彼らは Egenhofer および Herring の 9 Intersection Model をディメンション的に拡張しました。DE-9IM は 4 人の著者 (Clementini、Eliseo、Di Felice、および van Osstrom) の共同作業であり、 ″A Small Set of Formal Topological Relationships Suitable for End-User Interaction″ の中でこのモデルを発表しました。 , *Advances in Spatial Database—Third International Symposium. SSD '93*. LNCS 692. Pp. 277-295 の中でモデルを発表しました。9 Intersection model by M. J. Egenhofer and J. Herring (Springer-Verlag Singapore [1993]) は、″Categorizing binary topological relationships between regions, lines, and points in geographic databases,″ *Tech. Report, Department of Surveying Engineering*, University of Maine, Orono, ME 1991 で発表されました。

空間比較関数

比較関数には以下のものがあります。

- ST_Contains
- ST_Crosses
- ST_Disjoint
- ST_EnvIntersects

- ST_EqualCoordsys
- ST_Equals
- ST_EqualSRS
- ST_Intersects
- ST_MBRIntersects
- ST_Overlaps
- ST_Relate
- ST_Touches
- ST_Within

ある形状が別の形状を含むかどうかをチェックする関数

ST_Contains および ST_Within は両方共、2 つの形状を入力とし、一方の内部が他方の内部と交差するかどうかを判別します。分かりやすく言えば、ST_Contains は、特定の 1 番目の形状が 2 番目の形状を囲んでいるかどうか (つまり、1 番目が 2 番目を含んでいるかどうか) を判別します。 ST_Within は、1 番目の形状が完全に 2 番目の形状内にあるかどうか (つまり、1 番目が 2 番目の中にあるかどうか) を判別します。

ST_Contains

ある形状が別の形状に完全に包含されているかどうかを判別するには、ST_Contains を使用します。

ST_Contains は、2 番目の形状が 1 番目の形状に完全に包含されていれば値 1 を戻します。ST_Contains 関数は、ST_Within 関数と全く正反対の結果を戻します。

275 ページの図 44 は、ST_Contains の適用例を示しています。

- 複数ポイントに、ポイント、あるいは他の複数ポイントが包含されているということは、そのポイントの全体や 2 番目の複数ポイントに属する全ポイントが、1 番目の複数ポイントの中に入っているということを意味します。
- ポリゴンに複数ポイントが包含されているということは、その複数ポイントに属するすべてのポイントがポリゴンの境界上かポリゴンの内部にあるということを意味します。
- 折れ線にポイント、複数ポイント、あるいは他の折れ線が包含されているということは、複数ポイントや 2 番目の折れ線に属するすべてのポイントが 1 番目の折れ線の中に入っているということを意味します。
- ポリゴンにポイント、折れ線、あるいは他のポリゴンが包含されているということは、ポリゴンや折れ線、2 番目のポリゴンが 1 番目のポリゴンの内部にあるということを意味します。

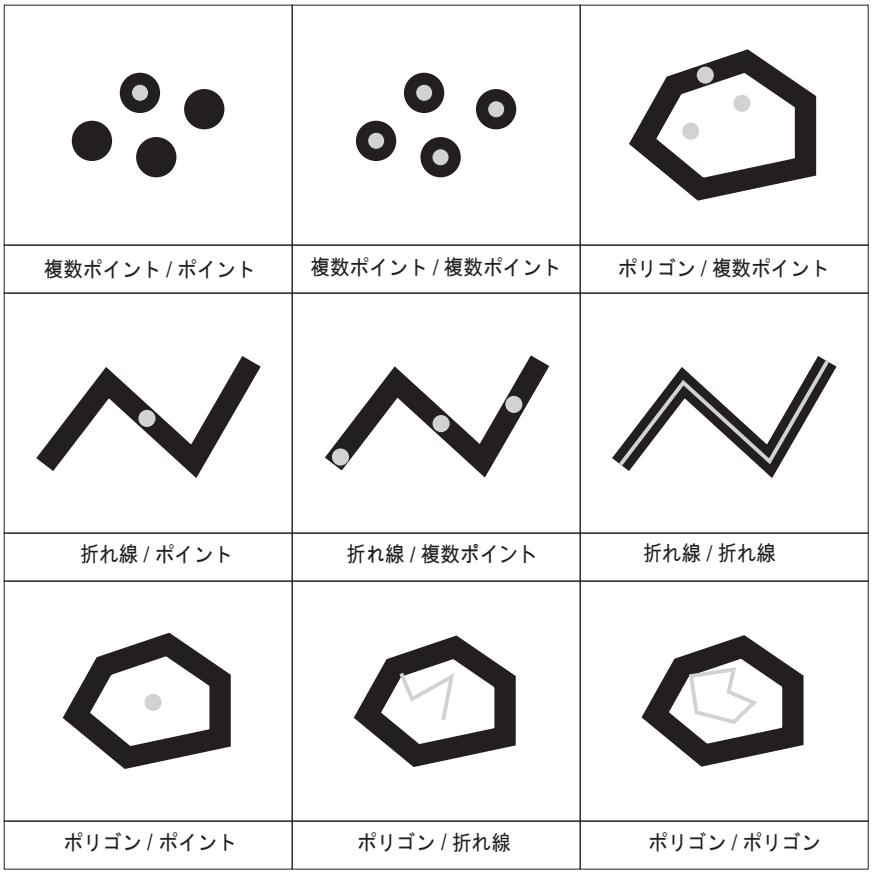


図 44. ST_Contains： 黒い形状は形状 a を表し、グレーの形状は形状 b を表します。すべてのケースにおいて、形状 a は形状 b を完全に包含しています。

ST_Contains 関数のパターン・マトリックスは、両方の形状の内部は交差しなければならないこと、および 2 番目 (形状 *b*) の内部または境界は 1 番目 (形状 *a*) の外部と交差してはならないことを表しています。アスタリスク (*) は、形状の該当部分に交差があっても特に問題がないということを意味します。

表 42. ST_Contains のマトリックス

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 内部	T	*	*
形状 a 境界	*	*	*
形状 a 外部	F	F	*

ST_Within

ある形状が完全に別の形状内にあるかどうかを判別するには、ST_Within を使用します。

ST_Within は、1 番目の形状が完全に 2 番目の形状内にある場合、値 1 を戻します。ST_Within は、ST_Contains と正反対の結果を戻します。

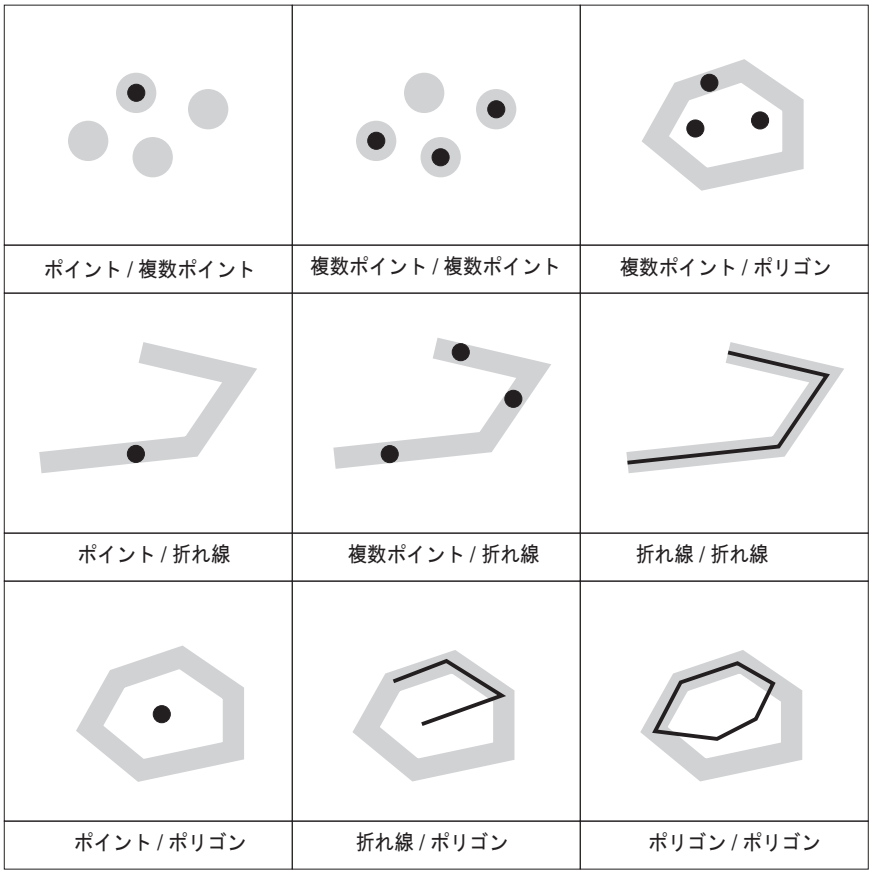


図 45. ST_Within

ST_Within 関数のパターン・マトリックスは、両方の形状の内部が交差しなければならないこと、および 1 次 (形状 *a*) の内部または境界は 2 次 (形状 *b*) の外部と交差してはならないことを表しています。アスタリスク (*) は、該当部分の交差は問題がないということを意味します。

表 43. ST_Within のマトリックス

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 内部	T	*	F
形状 a 境界	*	*	F
形状 a 外部	*	*	*

図 45 は、ST_Within の適用例を示しています。

- ポイントは、内部が 2 番目の複数ポイントの点のうちの 1 つと交差していれば、複数ポイントの中にあることになる。
- 複数ポイントは、すべてのポイントの内部が他の複数ポイントと交差していれば、他の複数ポイントの中にあることになる。
- 複数ポイントは、すべての点がポリゴンの境界上か内部にあれば、ポリゴンの中にあるということになる。

- ポイントは、全体が折れ線の内部にあれば、折れ線の中にあることになる。276 ページの図 45 では、内部が折れ線と交差していないため、ポイントは折れ線の中にあることにはならないが、複数ポイントは、すべてのポイントが折れ線の内部と交差しているため、折れ線の中にあることになる。
- 1 番目の折れ線のすべてのポイントが 2 番目の折れ線と交差している時には、1 番目の折れ線は 2 番目の折れ線の中にあることになる。
- ポイントは、ポリゴンの境界、あるいは内部と交差していないため、ポリゴンの中にあることにならない。
- 折れ線は、すべてのポイントがポリゴンの境界、あるいは内部と交差している時には、ポリゴンの中にあることになる。
- ポリゴンは、すべてのポイントが他のポリゴンの境界、あるいは内部と交差している時には、他のポリゴンの中にあることになる。

形状と形状が交差するかどうかをチェックする関数

ST_Intersects、ST_Crosses、ST_Overlaps、および ST_Touches はどれも、1 つの形状が他の形状と交差するかどうかを判別します。これらは、主としてテストする交差の範囲が異なります。

- ST_Intersects は、与えられた 2 つの形状が以下に述べる 4 つの条件の 1 つに合致するかどうかを判別するテストをします。形状の内部が交差すること。形状の境界が交差すること。1 番目の形状の境界が 2 番目の形状の内部と交差すること。または、1 番目の形状の内部が 2 番目の形状の境界と交差すること。
- ST_Crosses は、異なるディメンションの形状の交差を分析するために使用されます。ただし、例外が 1 つあります。それは、折れ線の交差も分析できるということです。すべてのケースにおいて、交差の場所自体が形状と見なされます。ST_Crosses の場合、この形状が、交差する形状より小さいディメンションの形状でなければなりません (あるいは、両者が折れ線である場合、交差の場所が折れ線よりも小さなディメンションの形状である必要があります)。例えば、折れ線およびポリゴンのディメンションは、それぞれ 1 および 2 です。このような 2 つの形状が交差し、交差の場所が線形 (ポリゴンに沿った折れ線のパス) である場合、その場所自体を折れ線であると見なすことができます。折れ線のディメンション (1) はポリゴンのディメンション (2) よりも小さいので、ST_Crosses は交差を分析した後、値 1 を戻します。
- 入力として ST_Overlaps に与えられる形状は同じディメンションの形状でなければなりません。 ST_Overlaps の場合、これらの形状が一部分オーバーラップし、それらの形状と同じディメンションである新規の形状 (オーバーラップの領域) を形成する必要があります。
- ST_Touches は、2 つの形状の境界が交差するかどうかを判別します。

ST_Intersects

2 つの形状が交差するかどうかを判別するには、ST_Intersects を使用します。

ST_Intersects は、交差の結果が空の集合にならない場合、値 1 を戻します。ST_Intersects は、ST_Disjoint と正反対の結果を戻します。

ST_Intersects 関数は、以下のパターン・マトリックスのいずれかの条件が TRUE を戻す場合、1 を戻します。

表 44. ST_Intersects のマトリックス (1): ST_Intersects 関数は、両方の形状の内部が交差する場合、1 を戻します。

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 境界	*	*	*
形状 a 内部	T	*	*
形状 a 外部	*	*	*

表 45. ST_Intersects のマトリックス (2): ST_Intersects 関数は、1 番目の形状の境界が 2 番目の形状の境界に交差する場合、1 を戻します。

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 境界	*	*	*
形状 a 内部	*	T	*
形状 a 外部	*	*	*

表 46. ST_Intersects のマトリックス (3): ST_Intersects 関数は、1 番目の形状の境界が 2 番目の形状の内部に交差する場合、1 を戻します。

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 境界	T	*	*
形状 a 内部	*	*	*
形状 a 外部	*	*	*

表 47. ST_Intersects のマトリックス (4): ST_Intersects 関数は、いずれかの形状の境界が交差する場合、1 を戻します。

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 境界	*	T	*
形状 a 内部	*	*	*
形状 a 外部	*	*	*

ST_Crosses

ある形状が別の形状と交わっているかどうかを判別するには、ST_Crosses を使用します。

ST_Crosses は 2 つの形状を入力とし、以下の場合に値 1 を戻します。

- 交差が、結果としてディメンションが元の形状の最大ディメンションよりも小さくなるような形状になる場合。
- 交差の集合が、元の両方の形状の内側にある場合。

ST_Crosses は、1 番目の形状が面または複数面である場合、あるいは 2 番目の形状がポイントまたは複数ポイントである場合、NULL を戻します。他のすべての組み合わせについては、ST_Crosses は値 1 (2 つの形状が交差することを示す) または値 0 (2 つの形状は交差しない) のいずれかを戻します。

以下の図は、複数ポイントと折れ線、折れ線と折れ線、複数ポイントとポリゴン、折れ線とポリゴンが交差している状態を示したものです。 4 件のケースのうち 3

件では、形状 b が形状 a と交わっています。 4 件目のケースでは、複数ポイントはポリゴンの線と交わってはいませんが、形状 b のポリゴンの領域に触れています。

黒の形状は形状 a を表し、グレーの形状は形状 b を表します。

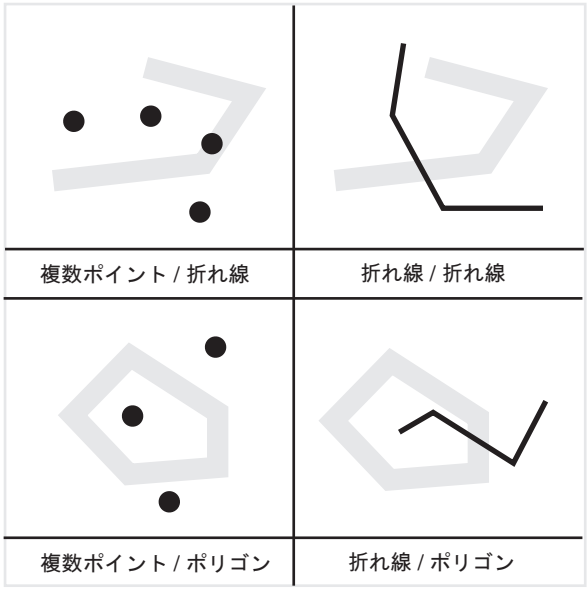


図 46. ST_Crosses

表 48 のパターン・マトリックスは、1 番目の形状がポイントまたは複数ポイントである場合、あるいは 1 番目の形状が曲線または複数曲線であって、2 番目の形状が面である場合に適用されます。このマトリックスは、内部が交差していなければならないこと、および 1 次 (形状 *a*) の内部が 2 次 (形状 *b*) の外部と交差していなければならないことを表します。

表 48. ST_Crosses のマトリックス (1)

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 境界	*	*	*
形状 a 内部	T	*	T
形状 a 外部	*	*	*

表 49 のパターン・マトリックスは、1 番目および 2 番目の両方の形状が曲線または複数曲線である場合に適用されます。0 は、内部の交差がポイント (0 ディメンション) でなければならないことを示しています。この交差のディメンションが 1 (折れ線における交差) である場合、ST_Crosses 関数は値 0 (形状は交差しないことを示す) を戻します。ただし、ST_Overlaps 関数は値 1 (形状はオーバーラップすることを示す) を戻します。

表 49. ST_Crosses のマトリックス (2)

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 境界	*	*	*
形状 a 内部	0	*	*
形状 a 外部	*	*	*

ST_Overlaps

同じディメンションの 2 つの形状がオーバーラップするかどうかを判別するには、ST_Overlaps を使用します。

ST_Overlaps は、同じディメンションの 2 つの形状を比較します。それらの交差の集合が、両者とは異なる形状 (ディメンションは同じ) になる場合、値 1 を戻します。

黒の形状は形状 *a* を表し、グレーの形状は形状 *b* を表します。すべてのケースにおいて、両方の形状は同じディメンションで、一方が他方と一部分オーバーラップしています。オーバーラップのエリアは新規の形状です。このエリアは形状 *a* および *b* と同じディメンションを持っています。

以下の図は、形状のオーバーラップを示しています。ポイント、折れ線、ポリゴンのオーバーラップの例です。ポイントの場合は、ポイント全体が重なり合っています。折れ線の場合は、線が部分的に重なり合っています。ポリゴンの場合は、領域が部分的に重なり合っています。

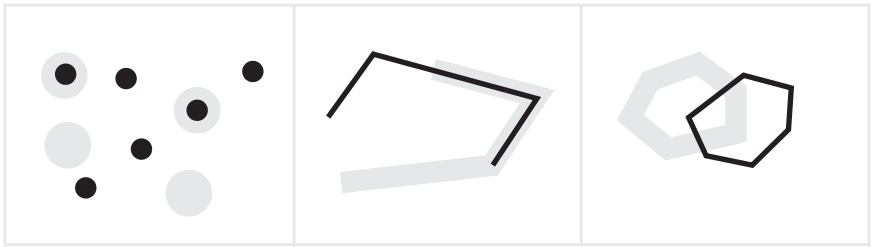


図 47. ST_Overlaps

表 50 のパターン・マトリックスは、1 番目および 2 番目の両方の形状が、ポイント、複数ポイント、面、または複数面である場合に適用されます。ST_Overlaps は、各形状の内部が他の形状の内部および外部と交差する場合、値 1 を戻します。

表 50. ST_Overlaps のマトリックス (1)

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 境界	*	*	*
形状 a 内部	T	*	T
形状 a 外部	T	*	*

表 51 のパターン・マトリックスは、1 番目および 2 番目の両方の形状が曲線または複数曲線である場合に適用されます。この場合、形状の交差は、結果としてディメンションが 1 の形状 (別の曲線) にならなければなりません。内部の交差のディメンションが 0 である場合、ST_Overlaps は値 0 (形状はオーバーラップしないことを示す) を戻します。ただし、ST_Crosses 関数は値 1 (形状は交差することを示す) を戻します。

表 51. ST_Overlaps のマトリックス (2)

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 境界	*	*	*
形状 a 内部	1	*	T

表 51. ST_Overlaps のマトリックス (2) (続き)

	形状 b 内部	形状 b 境界	形状 b 外部
形状 a 外部	T	*	*

ST_Touches

ST_Touches は、両方の形状に共通のすべてのポイントが境界上にのみある場合、値 1 を戻します。

形状の内部は、互いに交差していてはいけません。少なくとも、1 つの形状は曲線、面、複数曲線、または複数面でなければなりません。

黒の形状は形状 a を表し、グレーの形状は形状 b を表します。すべてのケースにおいて、形状 b の境界が形状 a と交差しています。形状 b の内部は、形状 a から独立した状態を保っています。

以下の図は、ポイントと折れ線、折れ線と折れ線、ポイントとポリゴン、複数ポイントとポリゴン、折れ線とポリゴンといった形状が接触している状態を示しています。

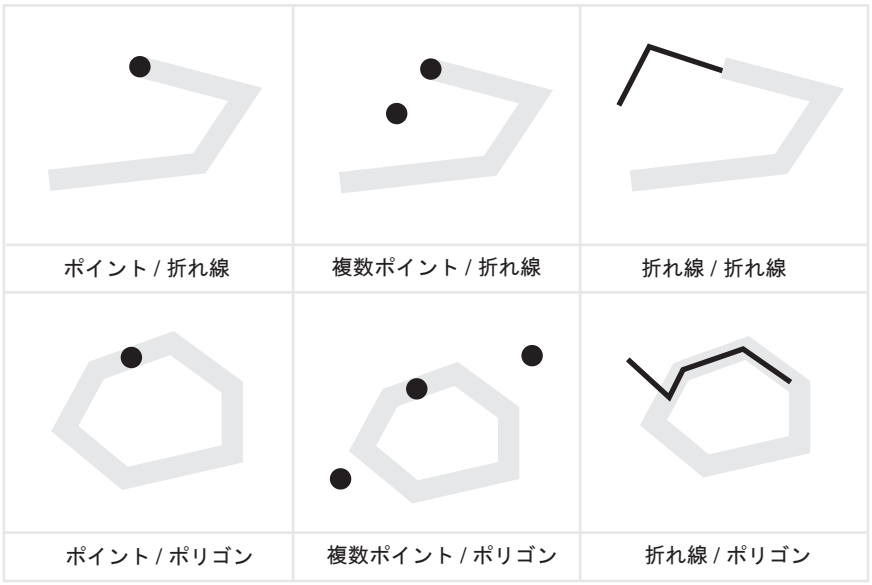


図 48. ST_Touches

形状の内部が交差せずに、いずれかの形状の境界が他方の内部または境界に交差する場合、ST_Touches 関数が 1 を戻すことをパターン・マトリックスは示しています。

表 52. ST_Touches のマトリックス (1)

	形状 b 内部	形状 b 境界	形状 b 外部
形状 a 境界	*	*	*
形状 a 内部	F	T	*
形状 a 外部	*	*	*

表 53. *ST_Touches* のマトリックス (2)

	形状 b 内部	形状 b 境界	形状 b 外部
形状 a 境界	T	*	*
形状 a 内部	F	*	*
形状 a 外部	*	*	*

表 54. *ST_Touches* のマトリックス (3)

	形状 b 内部	形状 b 境界	形状 b 外部
形状 a 境界	*	T	*
形状 a 内部	F	*	*
形状 a 外部	*	*	*

形状のエンベロープを比較する関数

ST_EnvIntersects と ST_MBRIntersects は、1 つの形状の最小外接長方形が、もう 1 つの形状の最小の外接長方形と交差するかどうかを判別するという点において、似ています。このような長方形は従来からエンベロープ と呼ばれてきました。複数ポリゴン、ポリゴン、複数折れ線、および湾曲折れ線は、それらのエンベロープの側面に接しています。水平折れ線、垂直折れ線、およびポイントは、それらのエンベロープよりも若干小さいです。ST_EnvIntersects は、形状のエンベロープが交差するかどうかを判別するテストをします。

形状を収めることのできる最小の長方形エリアのことを最小外接長方形 (MBR) と呼びます。複数ポリゴン、ポリゴン、複数折れ線、および湾曲折れ線を囲むエンベロープは、実際は MBR です。しかし、水平折れ線、垂直折れ線、およびポイントを囲むエンベロープは MBR ではありません。なぜなら、これらのエンベロープは、これら後者の形状が収まる最小のエリアを構成していないからです。これら後者の形状は、定義可能なスペースを占有しないので、MBR をもつことができません。それにもかかわらず、これらの形状がそれ自体の MBR と呼ばれる規則が採用されました。したがって、複数ポリゴン、ポリゴン、複数折れ線、および湾曲折れ線に関しては、ST_MBRIntersects は、ST_EnvIntersects がテストするのと同じ囲み長方形の交差をテストします。しかし、水平折れ線、垂直折れ線、およびポイントについては、ST_MBRIntersects はそれらの形状自体の交差をテストします。

ST_EnvIntersects

ST_EnvIntersects は、2 つの形状のエンベロープが交差する場合、値 1 を戻します。これは ST_Intersects (`ST_Envelope(g1),ST_Envelope(g2)`) を効率的にインプリメントする有用な関数です。

ST_MBRIntersects

ST_MBRIntersects は、2 つの形状の最小外接長方形 (MBR) が交差する場合、値 1 を戻します。

2 つのものが同一かどうかをチェックする関数

ST_EqualCoordsys

ST_EqualCoordsys は、2 つの座標システム定義が同一である場合、値 1 を戻します。

定義を比較するときに、ST_EqualCoordsys は大文字小文字、スペース、括弧、および浮動小数点表記の相違を無視します。

ST_Equals

ST_Equals は、2 つの形状が同一である場合、値 1 を戻します。

形状の定義に使用されるポイントの順序は、同一性のテストに関係しません。

以下に示した 6 つの例 (ポイント、複数ポイント、折れ線、複数ストリング、ポリゴン、複数ポリゴン) で、形状 a と形状 b は同じものです。

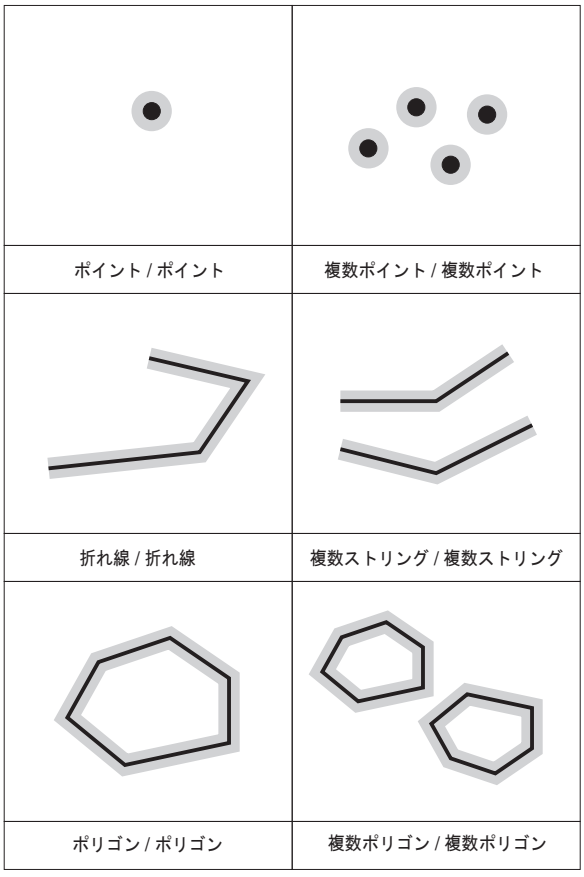


図 49. ST_Equals： 黒い形状は形状 a を表し、グレーの形状は形状 b を表します。すべてのケースにおいて、形状 a は形状 b と同じです。

表 55. 同一性のマトリックス： 同一性のための DE-9IM パターン・マトリックスは、内部が交差すること、およびどちらの形状の内部または境界のどの部分も、他方の外部と交差しないことを保証します。

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 境界	*	*	F
形状 a 内部	T	*	F
形状 a 外部	F	F	*

ST_EqualSRS

2 つの空間参照系が同一であるかどうかをテストするには、ST_EqualSRS を使用します。

ST_EqualSRS は、2 つの空間参照系が同一である場合、それらシステムのいずれかまたは両方の数字 ID が NULL でなければ、値 1 を戻します。

2 つの形状が交差していないかをチェックする関数

ST_Disjoint は、2 つの形状の交差が空の集合である場合、値 1 を戻します。この関数は、ST_Intersects が戻すものとは正反対のものを戻します。

以下の図は、いずれも、2 つの形状の境界がまったく交差していないという状態を示しています。

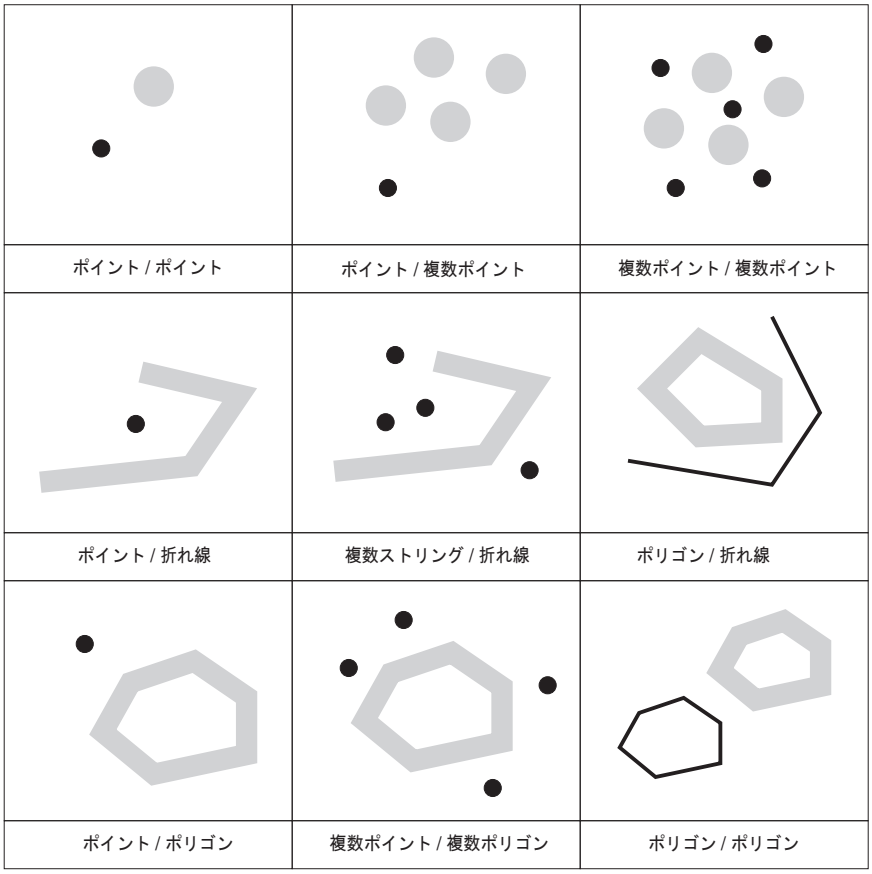


図 50. ST_Disjoint： 黒の形状は形状 a を表し、グレーの形状は形状 b を表します。すべてのケースにおいて、形状 a および形状 b は互いに交わっていません。

表 56. ST_Disjoint のマトリックス： このマトリックスは、単にどちらの形状の内部も境界も交差しないことを表しています。

	形状 b 内部	**形状 b 境界**	**形状 b 外部**
形状 a 境界	F	F	*
形状 a 内部	F	F	*
形状 a 外部	*	*	*

形状を DE-9IM パターン・マトリックス・ストリングと比較する関数

ST_Relate 関数は、2 つの形状を比較し、それらの形状が DE-9IM パターン・マトリックス・ストリングによって指定された条件に合致する場合は値 1 を戻します。そうでない場合は、関数は値 0 (ゼロ) を戻します。

形状のプロパティーについての情報を戻す関数

このセクションでは、形状のプロパティーについての情報を戻す空間処理関数を紹介しています。この情報は以下の項目に関係しています。

- 形状のデータ・タイプ
- 形状内の座標および指標
- リング、境界、エンベロープおよび最小外接長方形 (MBR)
- ディメンション
- 閉じている、空である、または単純であるという性質
- 形状集合内の基本形状
- 空間参照系

プロパティーによっては、それ自体で形状であるものがあります。例えば、表面の外部および内部リング、または曲線の開始および終了ポイントです。これらの形状は、このカテゴリーのいくつかの関数によって作成されます。他の種類の形状 (指定されたロケーションを囲むゾーンを表す形状など) を作成する関数は、別のカテゴリーに属します。『新規形状を生成する空間処理関数』と呼ばれる、この別のカテゴリーについては、このセクションの最後に記載されている該当するリンクまたは相互参照を参照してください。

データ・タイプ情報を戻す関数

ST_GeometryType は形状を入力パラメーターとし、その形状の動的タイプの完全修飾タイプ名を戻します。

座標および指標に関する情報を戻す関数

以下の関数は、形状内の座標および指標に関する情報を戻します。例えば、ST_X は指定されたポイント内の X 座標を戻します。また、ST_MaxX は形状内の最高 X 座標を戻し、ST_MinX は形状内の最低 X 座標を戻します。

これらの関数には以下のものがあります。

- ST_CoordDim
- ST_IsMeasured
- ST_IsValid
- ST_Is3D
- ST_M
- ST_MaxM
- ST_MaxX
- ST_MaxY
- ST_MaxZ
- ST_MinM
- ST_MinX

- ST_MinY
- ST_MinZ
- ST_X
- ST_Y
- ST_Z

ST_CoordDim

ST_CoordDim は、形状がもっている座標のタイプを表す値、および形状が指標も含んでいるかどうかを戻します。

この値は、*座標ディメンション*と呼ばれます。座標ディメンションは、ディメンションと呼ばれるプロパティーと同じものではありません。後者は、形状が幅または長さをもっているかどうかを示すものであって、特定のタイプの座標または指標を含んでいるかどうかを示すものではありません。

ST_IsMeasured

ST_IsMeasured は形状を入力パラメーターとし、与えられた形状が M 座標 (指標) を持つ場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

ST_IsValid

ST_IsValid は形状を入力パラメーターとし、それが有効な場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。形状は、構造化されたタイプのすべての属性が形状データの内部表記と整合していて、その内部表記が壊れていない場合にのみ、有効です。

ST_Is3D

ST_Is3d は形状を入力パラメーターとし、与えられた形状が Z 座標を持つ場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

ST_M

指標が、与えられたポイントと共に保管されている場合、ST_M はこのポイントを入力パラメーターとし、指標を戻します。

ST_MaxM

ST_MaxM は形状を入力パラメーターとし、その最大指標を戻します。

ST_MaxX

ST_MaxX は形状を入力パラメーターとし、その最大 X 座標を戻します。

ST_MaxY

ST_MaxY は形状を入力パラメーターとし、その最大 Y 座標を戻します。

ST_MaxZ

ST_MaxZ は形状を入力パラメーターとし、その最大 Z 座標を戻します。

ST_MinM

ST_MinM は形状を入力パラメーターとし、その最小指標を戻します。

ST_MinX

ST_MinX は形状を入力パラメーターとし、その最小 X 座標を戻します。

ST_MinY

ST_MinY は形状を入力パラメーターとし、その最小 Y 座標を戻します。

ST_MinZ

ST_MinY は形状を入力パラメーターとし、その最小 Z 座標を戻します。

ST_X

ST_X はポイントを入力パラメーターとし、そのポイントの X 座標を戻すことができます。

ST_Y

ST_Y はポイントを入力パラメーターとし、そのポイントの Y 座標を戻すことができます。

ST_Z

Z 座標が、与えられたポイントと共に保管されている場合、ST_Z はこのポイントを入力パラメーターとし、Z 座標を戻すことができます。

形状内の形状に関する情報を戻す関数

以下の関数は、形状内の形状に関する情報を戻します。いくつかの関数は形状内の特定のポイントを識別し、他の関数は集合内の基本形状数を戻します。

これらの関数には以下のものがあります。

- ST_Centroid
- ST_EndPoint
- ST_GeometryN
- ST_LineStringN
- ST_MidPoint
- ST_NumGeometries
- ST_NumLineStrings
- ST_NumPoints
- ST_NumPolygons
- ST_PointN
- ST_PolygonN
- ST_StartPoint

ST_Centroid

ST_Centroid は形状を入力パラメーターとし、形状の中心を戻します。形状の中心とは、与えられた形状の最小外接長方形の中心ポイントです。

ST_EndPoint

ST_Endpoint は曲線を入力パラメーターとし、その曲線の最後のポイントを戻します。

ST_GeometryN

ST_GeometryN は、形状の集合と 1 つの索引を入力パラメーターとし、集合の中から、索引で指定された形状を戻します。

ST_LineStringN

ST_LineStringN は、複数折れ線と索引を入力パラメーターとし、その索引で識別される折れ線を戻します。

ST_MidPoint

ST_MidPoint は曲線を入力パラメーターとし、曲線に沿って測定して、曲線の両端から等距離にある曲線上のポイントを戻します。

ST_NumGeometries

ST_NumGeometries は形状集合を入力パラメーターとし、その集合内にある形状の数を戻します。

ST_NumLineStrings

ST_NumLineStrings は複数折れ線を入力パラメーターとし、その中に含まれている折れ線の数を戻します。

ST_NumPoints

ST_NumPoints は、形状を入力パラメーターとし、その形状を定義するために使用されたポイントの数を戻します。例えば、形状がポリゴンであり、そのポリゴンを定義するために 5 つのポイントが使用されている場合、戻される数は 5 です。

ST_NumPolygons

ST_NumPolygons は複数ポリゴンを入力パラメーターとし、その中に含まれているポリゴンの数を戻します。

ST_PointN

ST_PointN は、折れ線または複数ポイントと索引を入力パラメーターとし、折れ線または複数ポイント内の、索引で指定されたポイントを戻します。

ST_PolygonN

ST_PolygonN は、複数ポリゴンと索引を入力パラメーターとし、索引で指定されたポリゴンを戻します。

ST_StartPoint

ST_StartPoint は曲線を入力パラメーターとし、その曲線の最初のポイントを戻します。

境界、エンベロープ、およびリングに関する情報を表示する関数

以下の関数は、形状の内部を外部から分割する境界、つまり形状そのものをその外部のスペースから分割する境界についての情報を戻します。例えば、ST_Boundary は、曲線の形で形状の境界を戻します。

これらの関数には以下のものがあります。

- ST_Boundary
- ST_Envelope
- ST_EnvIntersects
- ST_ExteriorRing
- ST_InteriorRingN
- ST_MBR
- ST_MBRIntersects
- ST_NumInteriorRing
- ST_Perimeter

ST_Envelope

ST_Envelope は形状を入力パラメーターとし、形状の周りのエンベロープを戻します。エンベロープはポリゴンとして表現される長方形です。

ST_EnvIntersects

ST_EnvIntersects は 2 つの形状を入力パラメーターとし、2 つの形状のエンベロープが交差する場合は 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

ST_ExteriorRing

ST_ExteriorRing はポリゴンを入力パラメーターとし、その外部リングを曲線として戻します。

ST_InteriorRingN

ST_InteriorRingN は、ポリゴンと索引を入力パラメーターとし、与えられた索引で指定された内部リングを折れ線として戻します。内部リングは、内部形状検査ルーチンにより定義された規則に従って編成されます。

ST_MBR

ST_MBR は形状を入力パラメーターとし、その最小外接長方形を戻します。

ST_MBRIntersects

ST_MBRIntersects は、2 つの形状の最小外接長方形 (MBR) が交差する場合、値 1 を戻します。

ST_NumInteriorRing

ST_NumInteriorRing はポリゴンを入力パラメーターとし、その内部リングの数を戻します。

ST_Perimeter

ST_Perimeter は、面または複数面、およびオプションで単位を入力パラメーターとして取り、特定の単位で測定された面または複数面の周囲の長さ (つまり、その境界の長さ) を戻します。

形状のディメンションに関する情報を戻す関数

以下の関数は、形状のディメンションに関する情報を戻します。例えば、ST_Area は、与えられた形状がカバーするエリアの大きさを報告します。

これらの関数には以下のものがあります。

- ST_Area
- ST_Dimension
- ST_Length

ST_Area

ST_Area は、形状およびオプションとして単位を入力パラメーターとして取り、与えられた形状がカバーするエリアを、指定された測定単位で戻します。

ST_Dimension

ST_Dimension は形状を入力パラメーターとし、そのディメンションを戻します。

ST_Length

ST_Length は、曲線または複数曲線および (オプションとして) 単位を入力パラメーターとし、与えられた曲線または複数曲線の長さを、与えられた測定単位で戻します。

形状が閉じているか、空か、または単純であるかを示す関数

以下の関数は、次のことを示します。

- 与えられた曲線または複数曲線が閉じているかどうか (つまり、曲線または複数曲線の開始ポイントおよび終了ポイントが同じであるかどうか)
- 与えられた形状が空であるかどうか (つまり、ポイントの欠けている状態)
- 曲線、複数曲線、または複数ポイントが単純であるかどうか (つまり、このような形状が一般的な構成をもっているかどうか)

ST_IsClosed

ST_IsClosed は曲線または複数曲線を入力パラメーターとし、与えられた曲線または複数曲線が閉じている場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

ST_IsEmpty

ST_IsEmpty は形状を入力パラメーターとし、与えられた形状が空の場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

ST_IsSimple

ST_IsSimple は形状を入力パラメーターとし、与えられた形状が単純な場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

形状の空間参照系を識別する関数

以下の関数は、形状に関連付けられている空間参照系を識別する値を戻します。さらに、関数 ST_SrsID は、形状を変更またはトランスフォームすることなしに、形状の空間参照系を変更することができます。

ST_SrsId (ST_SRID とも呼ばれる)

ST_SrsId (または ST_SRID) は、形状および (オプションとして) 空間参照系 ID を入力パラメーターとしてとります。この関数が何を戻すかは、入力パラメーターに何を指定したかによって異なります。

- 空間参照系 ID を指定した場合、関数は、指定した空間参照系に変更済みの形状を戻します。形状のトランスフォーメーションは行われません。
- 入力パラメーターに空間参照系 ID を指定しないと、与えられた形状の現行の空間参照系 ID が戻されます。

ST_SrsName

ST_SrsName は形状を入力パラメーターとし、与えられた形状を表現する空間参照系の名前を戻します。

既存の形状から新規形状を生成する関数

このセクションでは、既存の形状から新規形状を派生させる関数のカテゴリーを紹介します。このカテゴリーには、他の形状のプロパティーを表す形状を派生させる関数は含まれません。正確に言えば、以下のことを行う関数のカテゴリーです。

- 形状を他の形状に変換する
- スペースの構成を表す形状を作成する
- 複数の形状から個別の形状を派生させる
- 指標に基づいて形状を作成する
- 形状の修正版を作成する

ある形状を別の形状に変換する関数

以下の関数は、スーパータイプの形状を、対応するサブタイプの形状に変換することができます。例えば、ST_ToLineString 関数は、ST_Geometry タイプの折れ線を、ST_LineString の折れ線に変換することができます。これらの関数のいくつかは、基本形状および形状集合を、単一の形状集合に結合することもできます。例えば、ST_ToMultiLine は、折れ線と複数折れ線を、単一の複数折れ線に変換することができます。

ST_Polygon

ST_Polygon は、閉じた折れ線からポリゴンを構成することができます。折れ線は、ポリゴンの外部リングを定義します。

ST_ToGeomColl

ST_ToGeomColl は形状を入力パラメーターとし、これを形状の集合に変換します。

ST_ToLineString

ST_ToLineString は形状を入力パラメーターとし、これを折れ線に変換します。

ST_ToMultiLine

ST_ToMultiLine は形状を入力パラメーターとし、これを複数折れ線に変換します。

ST_ToMultiPoint

ST_ToMultiPoint は形状を入力パラメーターとし、これを複数ポイントに変換します。

ST_ToMultiPolygon

ST_ToMultiPolygon は形状を入力パラメーターとし、これを複数ポリゴンに変換します。

ST_ToPoint

ST_ToPoint は形状を入力パラメーターとし、これをポイントに変換します。

ST_ToPolygon

ST_ToPolygon は形状を入力パラメーターとし、これをポリゴンに変換します。

異なる空間構成を使用して新規形状を作成する関数

既存の形状を開始ポイントとして使用して、以下の関数は円形のエリアまたはスペースの他の構成を表す新規の形状を作成します。例えば、提案されている空港の中心を表すポイントが与えられたとして、ST_Buffer は提案される空港の範囲を表す面を、円形で作成することができます。

これらの関数には以下のものがあります。

- ST_Buffer

- ST_ConvexHull
- ST_Difference
- ST_Intersection
- ST_SymDifference

ST_Buffer

ST_Buffer 関数は、既存の形状から指定された半径だけ外側に広がる、新規形状を生成することができます。既存の形状がバッファーに入れられている場合、または集合のエレメントが接近しているために、集合の単一のエレメントの周りのバッファーがオーバーラップする場合は常に、新規の形状は面になります。ただし、バッファーが分かれている場合、結果として個別のバッファー面になります。この場合、ST_Buffer は複数面を戻します。

以下の図は、単一で、かつオーバーラップされたエレメントの周りのバッファーの例です。

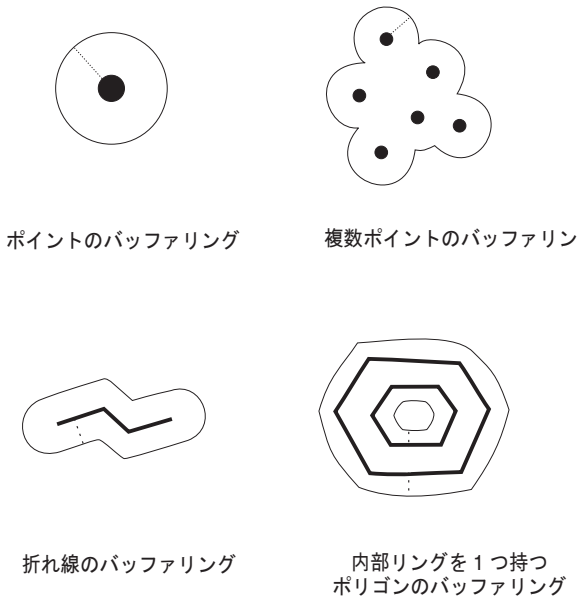


図 51. ST_Buffer

ST_Buffer 関数は正および負の両方の距離を受け入れます。ただし、ディメンションが 2 の形状 (面および複数面) のみが負のバッファーを使用します。元の形状のディメンションが 2 より小さい場合 (面でも複数面でもないすべての形状) は常に、バッファー距離の絶対値が使用されます。

一般的に、外部リングの場合、正のバッファー距離は、元の形状の中心から離れた面リングを生成します。負のバッファー距離は、中心へ向かう面または複数面リングを生成します。面または複数面の内部リングの場合、正のバッファー距離は、中心へ向かうバッファー・リングを生成し、負のバッファー距離は、中心から離れるバッファー・リングを生成します。

バッファリング・プロセスでは、オーバーラップする面をマージします。ポリゴンの最大内部幅の半分よりも大きい負の距離は、結果として空の形状になります。

ST_ConvexHull

ST_ConvexHull 関数は、凸面を形成する少なくとも 3 つの頂点を持つ任意の形状の凸包を戻します。頂点 は、形状内の X および Y 座標のペアです。凸包 は、与えられた頂点の集合内のすべての頂点によって形成することができる最小の凸面ポリゴンです。

以下の図には、凸包の例が 4 つ示してあります。最初の例では、アルファベットの c に似た不規則な形状が描かれています。凸包によって、c の開いている部分が閉じられたような格好になっています。4 つ目の例では、4 つのポイントがジグザグの線によって結ばれています。凸線は、一方の側では 2 番目のポイントと 4 番目のポイントの間に、もう一方の側では、1 番目のポイントと 3 番目のポイントの間に描かれています。

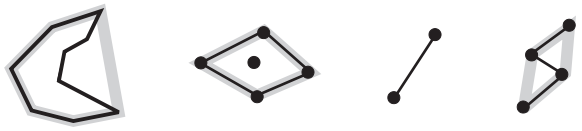


図 52. ST_ConvexHull

ST_Difference

ST_Difference は同じディメンションの 2 つの形状を入力としてとります。ST_Difference 関数は、1 番目の形状の、2 番目の形状が交差しない部分を戻します。この操作は、空間における論理 AND NOT 操作です。ST_Difference によって戻された形状の部分は、それ自体が形状、つまり入力となった形状と同じディメンションを持つ集合です。これらの 2 つの形状が等しい場合、つまり、それらが同じスペースを占める場合、戻される形状は空です。

各矢印の左側には、入力として ST_Difference に与えられた 2 つの形状があります。各矢印の右側には、ST_Difference の出力があります。1 番目の形状の一部に 2 番目の形状が交差する場合、1 番目の形状の交差していない部分が出力になります。入力として与えられた形状が等しい場合、出力は空の形状です (nil という用語で表されます)。

下の図は、ST_Difference の入力と出力の例です。例えば、入力がポイントで、ポイント a とポイント b が同じであれば、出力は NULL になります。ポイント a とポイント b が異なれば、出力は、両者の間の新しいポイントになります。入力がどちらも同じ形のポリゴンで、ポリゴン a の方が小さく、ポリゴン b の中に入っているという場合、出力は NULL になります。 2 つのポリゴンが重なり合っている場合、出力は重なったポリゴンの外側の端になります。

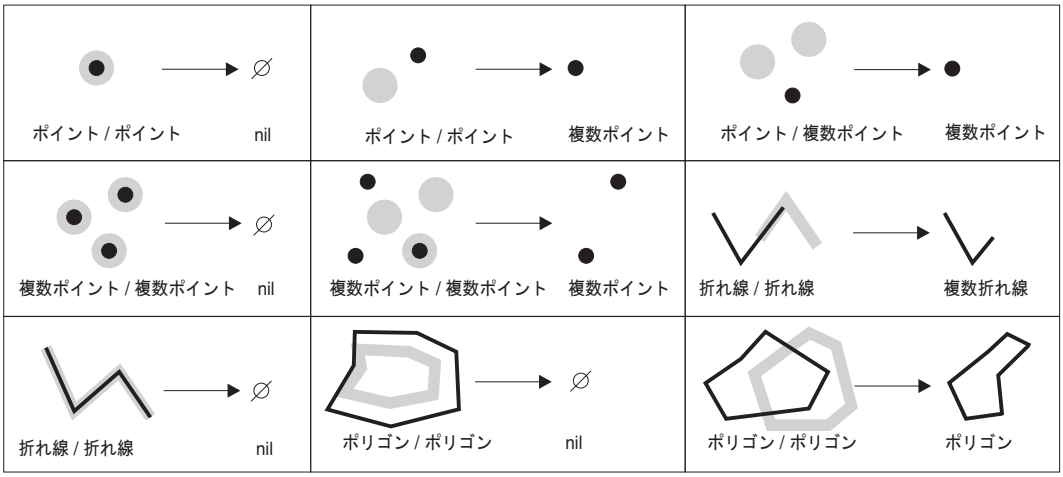


図 53. ST_Difference

ST_Intersection

ST_Intersection 関数は、2 つの与えられた形状の交差を定義する、形状として表される、ポイントの集合を戻します。

入力として ST_Intersection に与えられた形状が交差しない場合、またはそれらが交差して、それらの交差のディメンションが形状のディメンションよりも小さい場合、ST_Intersection は空の形状を戻します。

各矢印の左側には、入力として ST_Intersection に与えられた 2 つの交差する形状があります。各矢印の右側には、ST_Intersection の出力、つまり、左側の形状によって作成された交差を表す形状があります。

下の図は、ST_Intersection の出力の例です。入力として与えられた形状がどこで交差しているかという情報を戻します。例えば、形状 b が折れ線で形状 a が線上のポイントであった場合、出力は両者が重なる地点の複数ポイントになります。形状 a と形状 b が重なり合うポリゴンであれば、出力は、重なった部分だけからなる新たな複数ポリゴンになります。

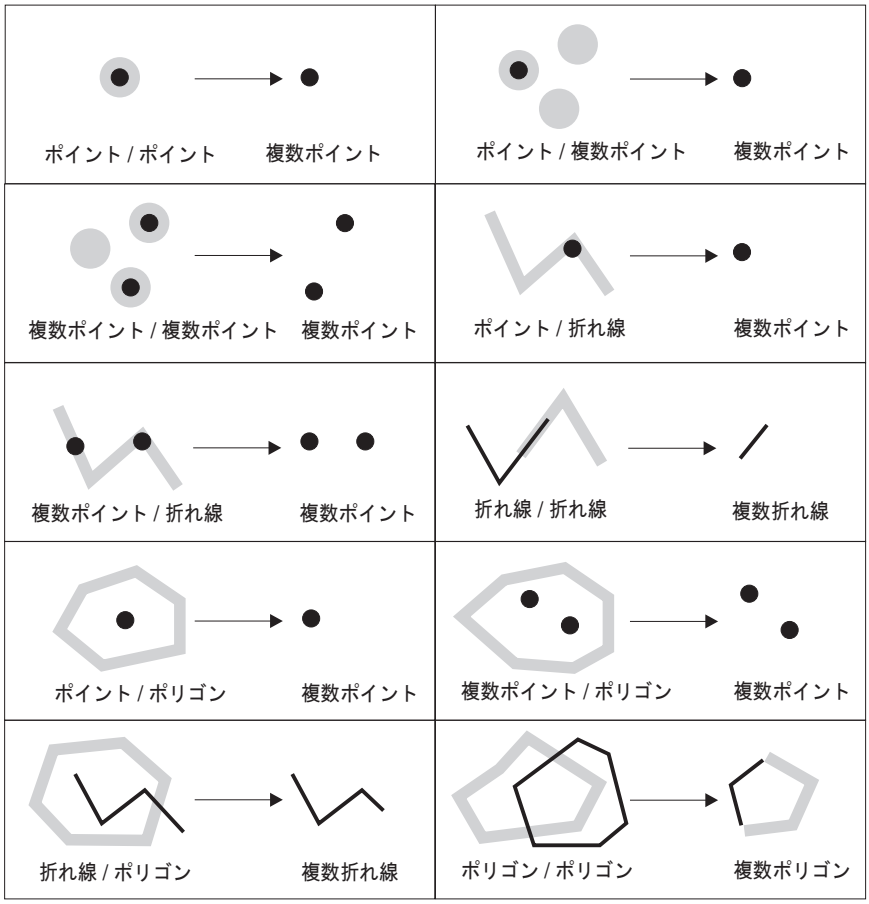


図 54. ST_Intersection

ST_SymDifference

ST_SymDifference 関数は、同じディメンションをもつ 2 つの交差する形状の対称差 (空間における論理 XOR 操作) を戻します。これらの形状が等しい場合、ST_SymDifference は空の形状を戻します。形状が等しくない場合は、形状の一方または両方の一部が、交差するエリアの外にあります。

複数の形状から 1 つの形状を派生させる関数

以下の関数は、複数の形状から個別の形状を派生させます。例えば、ST_Union は、2 つの形状を単一の形状に結合します。

MBR 集約

関数 ST_BuildMBRAggr および ST_GetAggrResult の組み合わせは、列の中のすべての形状の最小外接長方形を表す長方形を構成することによって、選択された列の中の形状の列を、単一の形状に集約します。集約を計算するときに、Z 座標と M 座標は破棄されます。

ST_Union

ST_Union 関数は、2 つの形状の共用体セットを戻します。

この操作は、空間における論理 OR です。2 つの形状は、同じディメンションの形状でなければなりません。ST_Union は、常に結果を集合として戻します。

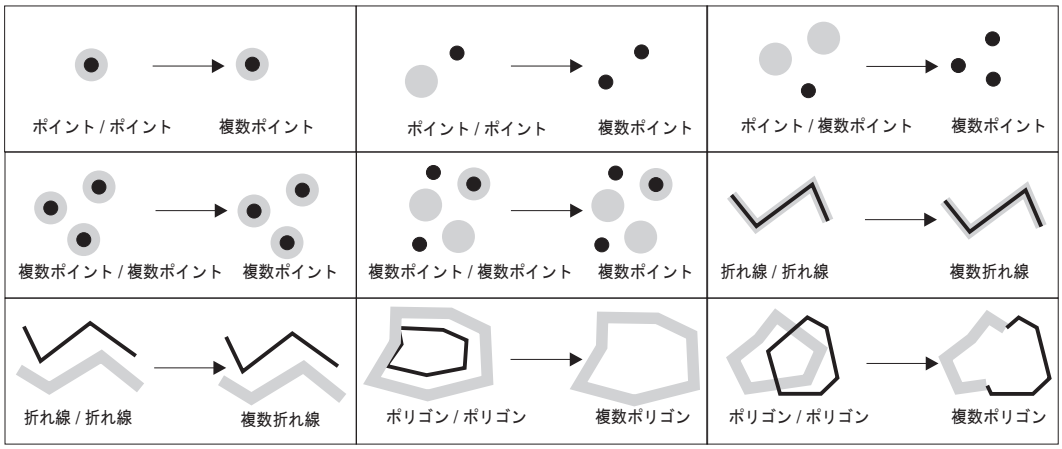


図 55. ST_Union

和集約

和集約は、ST_BuildUnionAggr と ST_GetAggrResult の関数を組み合わせたものです。この組み合わせは、表内の形状の列を和集約として集約し、1 つの形状にします。

指標を処理する関数

このセクションで説明されている関数は、指標値を持つ形状を処理します。これらの関数は、指標に基づく新規形状、またはある位置までの形状に沿った距離のいずれかを返します。

これらの関数には以下のものがあります。

ST_DistanceToPoint

ST_DistanceToPoint は、入力パラメーターとして単一曲線または複数曲線の形状と 1 つのポイント形状を取り、指定したポイントまでの曲線形状に沿った距離を返します。

この関数はメソッドとして呼び出すこともできます。

構文

```
►►—db2gse.ST_DistanceToPoint—(—curve-geometry—,—point-geometry—)————————►◄
```

パラメーター

curve-geometry
処理する形状を表す、ST_Curve タイプまたは ST_MultiCurve タイプまたはそのサブタイプの 1 つの値。

point-geometry
指定した曲線に沿ったポイントであるタイプ ST_Point の値。

戻りタイプ

DOUBLE

例

例 1

以下の SQL ステートメントにより、2 つの列を持つ SAMPLE_GEOMETRIES 表が作成されます。ID 列で各行が一意的に識別されます。 GEOMETRY ST_LineString 列にはサンプル形状が格納されます。

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

以下の SQL ステートメントにより、2 つの行が SAMPLE_GEOMETRIES 表に挿入されます。

```
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))
```

以下の SELECT ステートメントおよび対応する結果セットは、ST_DistanceToPoint 関数を使用して位置 (1.5, 15.0) にあるポイントまでの距離を検出する方法を示しています。

```
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.
```

ST_FindMeasure または ST_LocateAlong

ST_FindMeasure または ST_LocateAlong は形状と指標を入力パラメーターとし、指定された指標を含んでいる、指定された形状の指定された指標そのものをもつ形状部分の、複数ポイントまたは複数曲線を戻します。

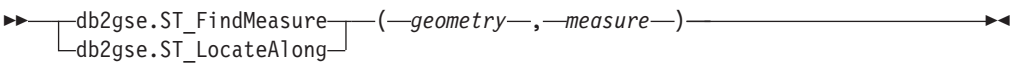
ポイントおよび複数ポイントについては、指定された指標をもっているすべてのポイントが戻されます。曲線、複数曲線、面、および複数面の場合、結果を計算するために補間が行われます。面および複数面の計算は、形状の境界に関して行われます。

ポイントおよび複数ポイントの場合、与えられた指標が見つからない場合は、空の形状が戻されます。他のすべての形状については、与えられた指標が形状内の最小

の指標より低い場合、または形状内の最高の指標より高い場合は、空の形状が戻されます。与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
M 座標 (指標) に *measure* が含まれる部分を検索する形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

measure
geometry の一部が結果に含まれていなければならない指標を示す、DOUBLE タイプの値。

戻りタイプ

db2gse.ST_Geometry

例

例 1

次の CREATE TABLE ステートメントは、SAMPLE_GEOMETRIES 表を作成します。SAMPLE_GEOMETRIES には、ID 列 (各行を一意的に識別する列)、および GEOMETRY ST_Geometry 列 (サンプルの形状を保管する列) の 2 つの列があります。

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id SMALLINT, geometry ST_GEOMETRY)
```

次の INSERT ステートメントは 2 つの行を挿入します。最初の行は折れ線であり、2 番目の行は複数ポイントです。

```
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))
```

例 2

次の SELECT ステートメントおよび対応する結果セットでは、ST_FindMeasure 関数を使用して指標が 7 のポイントを見つけています。最初の行は 1 つのポイントを戻します。しかし、2 番目の行は空のポイントを戻します。線形フィーチャー (0 より大きいディメンションを持つ形状) の場合、ST_FindMeasure はポイントを補間できますが、複数ポイントの場合、ターゲットの指標は正確に一致する必要があります。

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 7))
   AS varchar(45)) AS measure_7
FROM   sample_geometries
```

結果:

```
ID     MEASURE_7
------ --------------------------------------------
     1 POINT M ( 3.50000000 3.50000000 7.00000000)
     2 POINT EMPTY
```

例 3

次の SELECT ステートメントおよび対応する結果セットで、ST_FindMeasure 関数は 1 つのポイントと 1 つの複数ポイントを戻します。ターゲットの指標 6 は、ST_FindMeasure および複数ポイントのソース・データの両方の指標と一致します。

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 6))
   AS varchar(120)) AS measure_6
FROM   sample_geometries
```

結果:

```
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)
```

ST_MeasureBetween または ST_LocateBetween

ST_MeasureBetween または ST_LocateBetween は、形状および 2 つの M 座標 (指標) を入力パラメーターとし、その形状の、2 つの M 座標の間の切断されたパスまたはポイントのセットを表す部分を戻します。

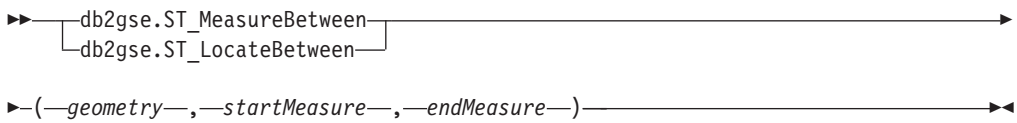
曲線、複数曲線、面、および複数面の場合、結果を計算するために補間が行われます。結果の形状は、与えられた形状の空間参照系で表現されます。

与えられた形状が面または複数面の場合、ST_MeasureBetween または ST_LocateBetween は形状の外部および内部リングに適用されます。与えられた形状のいずれの部分も与えられた M 座標で定義されたインターバルにない場合は、空の形状が戻されます。与えられた形状が NULL の場合は NULL が戻されます。

結果の形状が空でない場合、複数ポイントまたは複数折れ線のタイプが戻されます。

どちらの関数も、メソッドとして呼び出すことができます。

構文



パラメーター

geometry
指標値が *startMeasure* から *endMeasure* である部分がある形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

startMeasure
測定インターバルの下限を表すDOUBLE タイプの値。この値が NULL の場合、下限は適用されません。

endMeasure
測定インターバルの上限を表すDOUBLE タイプの値。この値が NULL の場合、上限は適用されません。

戻りタイプ

db2gse.ST_Geometry

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

形状の M 座標 (指標) はユーザーにより定義されます。指標は、測定したいものを何でも表現できる (例えば、高速道路の距離、温度、圧力、pH など) ので、千差万別です。

この例は、pH を測定して収集したデータを記録するための M 座標の使用を説明しています。研究者は特定の場所の高速道路に沿った土壌の pH を収集しています。通常の操作手順にしたがって、研究者は土壌サンプルを採取した場所ごとに必要な値 (採取した場所の X 座標、Y 座標、および測定した pH 値) を書き留めます。

```
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 ) )
```

土壌の酸性が 4 から 6 の間で変化するパスを見つけるには、研究者は次の SELECT ステートメントを使用します。

```
SELECT id, CAST( ST_AsText( ST_MeasureBetween( 4, 6) )
  AS VARCHAR(150) ) MEAS_BETWEEN_4_AND_6
  FROM sample_lines
```

結果:

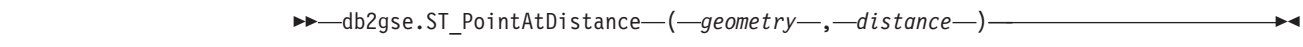
```
ID         MEAS_BETWEEN_4_AND_6
---------- ----------------------------------------------------
         1 LINESTRING M (3.00000000 4.33333300 4.00000000,
                         3.00000000 3.00000000 6.00000000,
                         4.00000000 4.00000000 6.00000000,
                         5.00000000 5.00000000 6.00000000)
```

ST_PointAtDistance

ST_PointAtDistance は、入力パラメーターとして単一曲線または複数曲線の形状と 1 つの距離を取り、曲線形状に沿った所定の距離にあるポイント形状を返します。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
処理する形状を表す、ST_Curve タイプまたは ST_MultiCurve タイプまたはそのサブタイプの 1 つの値。

distance
ポイントを見つけるための、形状に沿った距離であるタイプ DOUBLE の値。

戻りタイプ

db2gse.ST_Point

例

例 1

以下の SQL ステートメントにより、2 つの列を持つ SAMPLE_GEOMETRIES 表が作成されます。ID 列で各行が一意的に識別されます。 GEOMETRY ST_LineString 列にはサンプル形状が格納されます。

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

以下の SQL ステートメントにより、2 つの行が SAMPLE_GEOMETRIES 表に挿入されます。

```
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))
```

以下の SELECT ステートメントおよび対応する結果セットは、ST_PointAtDistance 関数を使用して行ストリングの開始から 15 座標単位の距離にあるポイントを検出する方法を示しています。

```
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.
```

既存の形状の修正フォームを作成する関数

以下の関数は、既存の形状の修正フォームを作成します。例えば、ST_AppendPoint 関数は、既存の曲線の拡張版を作成します。各バージョンには、既存の曲線の中のポイントと追加のポイントが組み込まれます。

これらの関数には以下のものがあります。

- ST_AppendPoint
- ST_ChangePoint
- ST_Generalize
- ST_M
- ST_PerpPoints
- ST_RemovePoint
- ST_X
- ST_Y
- ST_Z

ST_AppendPoint

ST_AppendPoint は曲線とポイントを入力パラメーターとし、与えられたポイントにより曲線を拡張します。

ST_ChangePoint

ST_ChangePoint は 1 つの曲線と 2 つのポイントを入力パラメーターとします。これは、与えられた曲線内で、1 番目のポイントと同じポイントをすべて 2 番目のポイントで置き換え、結果の曲線を戻します。

ST_Generalize

ST_Generalize は形状としきい値を入力パラメーターとし、形状の一般的特性を保持しつつ、ポイントの数を減らして、与えられた形状を表現します。Douglas-Peucker line-simplification (ダグラス・デッカーの線単純化) アルゴリズムを使用し、これにより、ポイントの並びを直線セグメントで置き換えることができるようになるまで、形状を定義する一連のポイントを繰り返し分割します。この線セグメント内では、定義されたポイントはすべて、与えられたしきい値を超えて直線セグメントから離れることはありません。Z および M 座標は単純化には考慮されません。

ST_M

与えられたポイントが指標に関連付けられていない場合、ST_M がそのポイントと共に保管する指標を提供することができます。ポイントに関連付けられた指標がある場合、ST_M はその指標を別の指標に置き換えることができます。

ST_PerpPoints

ST_PerpPoints は、曲線または複数曲線とポイントを入力パラメーターとし、その曲線または複数曲線上の与えられたポイントの垂直投影を戻します。与えられたポイ

ントと垂直ポイントの間の距離が最小のポイントが戻されます。2 つ以上のこのような垂直に投影されたポイントが、与えられたポイントから等距離にある場合、それらすべてのポイントが戻されます。

ST_RemovePoint

ST_RemovePoint は曲線とポイントを入力パラメーターとし、指定されたポイントと等しいポイントをすべて与えられた曲線から除去して戻します。与えられた曲線が Z または M 座標を持つ場合、ポイントも Z または M 座標を持つ必要があります。

ST_X

ST_X は、ポイントの X 座標を別の X 座標に置き換えることができます。

ST_Y

ST_Y は、ポイントの Y 座標を別の Y 座標に置き換えることができます。

ST_Z

与えられたポイントが Z 座標をもっていない場合、ST_Z はそのポイントに Z 座標を追加することができます。ポイントが Z 座標をもっている場合、ST_Z はその座標を別の Z 座標に置き換えることができます。

距離情報を戻す関数

ST_Distance は、2 つの形状およびオプションとして単位を入力パラメーターとして取り、1 番目の形状内の任意のポイントと 2 番目の形状内の任意のポイントとの間の最短距離を、指定された単位で戻します。

2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。

指定された 2 つの形状のいずれかが NULL または空の場合は、NULL が戻されます。

例えば、ST_Distance は、航空機が飛行する必要がある 2 つのロケーション間の最短距離を報告できます。 306 ページの図 56 はこの情報を示しています。

図は、米国の地図にロサンゼルスとシカゴを結ぶ直線をつけたものです。

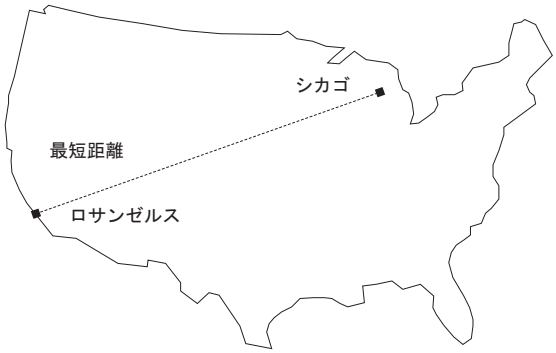


図 56. 2 つの都市の間の最短距離： ST_Distance は、ロサンゼルスとシカゴのロケーションの座標を入力とし、それらのロケーション間の最短距離を表す値を戻すことができます。

索引情報を戻す関数

ST_GetIndexParms は、空間インデックスの ID または空間列の ID のいずれかを入力パラメーターとし、索引または空間列の索引を定義するために使用されるパラメーターを戻します。追加のパラメーター番号が指定されると、その番号によって識別されるパラメーターのみが戻されます。

座標システム間の変換

ST_Transform は、形状および空間参照系 ID を入力パラメーターとし、指定された空間参照系で表現されるようにその形状をトランスフォームします。異なる座標システムの間で投影および変換が行われ、形状の座標はそれに従って調整されます。

第 23 章 空間処理関数: 構文およびパラメーター

このセクションでは、次のセクションで記述されている空間処理関数を紹介しています。ここでは、すべての、またはほとんどの空間処理関数に共通な、いくつかの係数を説明しています。ここでは関数はアルファベット順に記述されています。

空間処理関数: 考慮事項および関連データ・タイプ

このセクションには、空間処理関数をコーディングする場合に必要な情報が記載されています。その情報は以下のものです。

- 考慮する要因: 空間処理関数が属するスキーマを指定するための要件および、いくつかの関数はメソッドとして呼び出すことができるという事実。
- 別の空間処理関数から戻された形状のタイプを空間処理関数が処理できない場合に、どのように対処するか。
- どの関数が各空間データの値を入力として取るかを示す表

空間処理関数を使用する場合は、以下の要因に注意してください。

- 空間処理関数を呼び出すには、その名前を、空間処理関数が属するスキーマ名 (DB2GSE) で修飾する必要があります。これを行う 1 つの方法は、関数を参照する SQL ステートメント内でスキーマを明示的に指定することです。例えば、次のように指定します。

  ```
  SELECT db2gse.ST_Relate (g1, g2, 'T*F**FFF2') EQUALS FROM relate_test
  ```

  別の方法として、関数を呼び出すたびにスキーマを指定することを避けるために、CURRENT FUNCTION PATH 特殊レジスターに DB2GSE を追加することができます。この特殊レジスターの現行設定値を得るには、次の SQL コマンドを入力します。

  ```
  VALUES CURRENT FUNCTION PATH
  ```

  CURRENT FUNCTION PATH 特殊レジスターを DB2GSE に更新するには、次の SQL コマンドを発行します。

  ```
  set CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
  ```

- いくつかの空間処理関数はメソッドとして呼び出すことができます。例えば、次のコーディングで ST_Area は最初に関数として呼び出され、次にメソッドとして呼び出されています。この両方とも、ST_Area は、STORES という名前の表の SALES_ZONE 列に保管されている、ID が 10 のポリゴンを操作するようにコーディングされています。呼び出されると、ST_Area はポリゴンが表す実際の地形 (販売ゾーン NO. 10) を戻します。

  ST_Area は次のように関数として呼び出されます。

  ```
  SELECT ST_Area(sales_zone)
  FROM   stores
  WHERE  id = 10
  ```

  方式として呼び出される ST_Area

© Copyright IBM Corp. 1998, 2009

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

関数 ST_BuildMBRAggr および ST_BuildUnionAggr について、 『MBR 集約』および『和集約』でそれぞれ説明しています。

考慮する要因

ST_Geometry の値をサブタイプの値として扱う

静的タイプがスーパータイプである形状を空間処理関数が戻し、その形状をこのスーパータイプに従属するタイプの形状のみを受け付ける関数に渡すと、コンパイル時に例外が起こります。

例えば、ST_Union 関数の出力パラメーターの静的タイプは、すべての空間データのスーパータイプである ST_Geometry です。ST_PointOnSurface 関数の静的入力パラメーターは、ST_Geometry の 2 つのサブタイプである ST_Polygon または ST_MultiPolygon にすることができます。DB2® が ST_Union から戻された形状を ST_PointOnSurface に渡そうとすると、DB2 は以下のコンパイル時に例外を起こします。

```
SQL00440N No function by the name "ST_POINTONSURFACE"
having compatible arguments was found in the function
path.    SQLSTATE=42884
```

このメッセージは、ST_Geometry の入力パラメーターを持つ、ST_PointOnSurface という名前の関数を DB2 が見つけられなかったことを示します。

スーパータイプのサブタイプのみを受け付ける関数にスーパータイプの形状を渡すには、TREAT 演算子を使用します。前述のように、ST_Union は ST_Geometry の静的タイプの形状を戻します。これは、ST_Geometry の動的サブタイプの形状を戻すこともできます。ここで例えば、ST_MultiPolygon の動的タイプを持つ形状を戻すとします。この場合、TREAT 演算子は、この形状を静的タイプ ST_MultiPolygon で使用することを要求します。これは ST_PointOnSurface の入力パラメーターのデータ・タイプの 1 つと一致します。ST_Union が ST_MultiPolygon 値を戻さないと、DB2 はランタイム例外を起こします。

関数がスーパータイプの形状を戻す場合、TREAT 演算子は通常、この形状をこのスーパータイプのサブタイプと見なすように DB2 に伝えます。 ただしこの操作は、サブタイプが、形状が渡される関数の入力パラメーターとして定義された静的サブタイプと一致するか、またはこれに従属する場合にのみ成功する、ということに注意してください。この条件を満たさない場合、DB2 はランタイム例外を起こします。

別の例を考えてみましょう。例えば、穴を持たないポリゴンの境界上のあるポイントに対する垂直のポイントを知りたいとします。ST_Boundary 関数を使用して、ポリゴンから境界を導き出します。ST_Boundary の静的出力パラメーターは ST_Geometry ですが、ST_PerpPoints は ST_Curve 形状を受け付けます。すべてのポリゴンは、境界として折れ線 (これは曲線でもある) を持ち、また折れ線のデータ・タイプ (ST_LineString) は ST_Curve に従属するため、次の操作は ST_Boundary から戻された ST_Geometry ポリゴンを ST_PerpPoints に渡します。

```
SELECT ST_AsText(ST_PerpPoints(TREAT(ST_Boundary(polygon) as ST_Curve)),
       ST_Point(30.5, 65.3, 1)))
FROM   polygon_table
```

ST_Boundary と ST_PerpPoints を関数として呼び出す代わりに、これらをメソッドとして呼び出すことができます。これを行うには、次のようにコーディングします。

```
SELECT TREAT(ST_Boundary(polygon) as ST_Curve)..
       ST_PerpPoints(St_Point(30.5, 65.3, ))..ST_AsText()
FROM   polygon_table
```

入力タイプ別の空間処理関数のリスト

310 ページの表 57 は、受け入れ可能な入力のタイプにしたがって、空間処理関数をリストしています。

重要: 他の個所で述べられているように、空間データは ST_Geometry をルートとする階層を形成しています。 DB2 Spatial Extender の資料において、この階層内のスーパータイプの値を関数の入力として使用できることが示されている場合は、このスーパータイプのすべてのサブタイプの値もその関数の入力として使用することができます。

例えば、310 ページの表 57 の最初の項目には、ST_Area および他の多くの関数が ST_Geometry データ・タイプの値を入力にできることが示されています。したがって、これらの関数への入力は、ST_Geometry の任意のサブタイプ (ST_Point、ST_Curve、 ST_LineString など) の値にすることもできます。

表 57. 入力タイプ別の空間処理関数のリスト

入力パラメーターのデータ・タイプ	関数
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 または ST_LocateAlong
	ST_Generalize
	ST_GeometryType

表 57. 入力タイプ別の空間処理関数のリスト (続き)

入力パラメーターのデータ・タイプ	関数
ST_Geometry (続き)	
	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 または ST_LocateBetween
	ST_MinM
	ST_MinX
	ST_MinY
	ST_MinZ
	ST_NumPoints
	ST_Overlaps
	ST_Relate
	ST_SRID または 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

表 57. 入力タイプ別の空間処理関数のリスト (続き)

入力パラメーターのデータ・タイプ	関数
ST_Curve	
	ST_AppendPoint
	ST_ChangePoint
	ST_EndPoint
	ST_IsClosed
	ST_IsRing
	ST_Length
	ST_MidPoint
	ST_PerpPoints
	ST_RemovePoint
	ST_StartPoint
ST_LineString	
	ST_PointN
	ST_Polygon
ST_Surface	
	ST_Perimeter
	ST_PointOnSurface
ST_GeomCollection	
	ST_GeometryN
	ST_NumGeometries
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

EnvelopesIntersect

EnvelopesIntersect は、以下の 2 つのタイプの入力パラメーターを受け入れます。

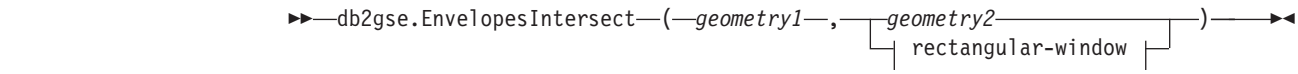
- 2 つの形状

  EnvelopesIntersect は、最初の形状のエンベロープが 2 番目の形状のエンベロープと交差する場合に 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

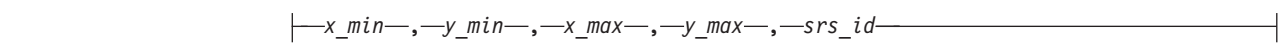
- 形状、長方形ウィンドウの左下隅と右上隅を定義するタイプ DOUBLE の 4 つの座標値、および空間参照系 ID。

  EnvelopesIntersect は、最初の形状のエンベロープがタイプ DOUBLE の 4 つの値で定義されたエンベロープと交差する場合は、 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

構文



rectangular-window:



パラメーター

geometry1
geometry2 またはタイプ DOUBLE の 4 つの値で定義された長方形ウィンドウのエンベロープとの交差をテストするエンベロープの、形状を表すST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2
geometry1 のエンベロープとの交差をテストするエンベロープの、形状を表す ST_Geometry タイプまたはそのサブタイプの 1 つの値。

x_min
エンベロープの最小 X 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

測地データに関しては、以下の条件が適用されます。

- *x_min* は、–180 度から 180 度までの間の経度値でなければならない。
- *x_min* は、エンベロープが 180 度子午線とオーバーラップする場合、*x_max* より大きくなくてはならない。

y_min
エンベロープの最小 Y 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

測地データに関しては、以下の条件が適用されます。

- *y_min* は、–90 度から 90 度までの間の緯度値でなければならない。
- *y_min* の値は *y_max* より小さくなくてはならない。

x_max
エンベロープの最大 X 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

測地データに関しては、以下の条件が適用されます。

- *x_max* は、–180 度から 180 度までの間の経度値でなければならない。
- *x_max* の値は、エンベロープが 180 度子午線とオーバーラップする場合、*x_min* より小さくなくてはならない。

y_max
エンベロープの最大 Y 座標値を指定します。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは DOUBLE です。

測地データに関しては、以下の条件が適用されます。

- *y_max* は、–90 度から 90 度までの間の緯度値でなければならない。
- *y_max* の値は *y_min* より大きくなくてはならない。

srs_id
空間参照系を一意的に識別します。空間参照系 ID は、形状パラメーターの空間参照系 ID と一致していなければなりません。このパラメーターには NULL でない値を指定する必要があります。

このパラメーターのデータ・タイプは INTEGER です。

戻りタイプ

INTEGER

例

この例は、郡を表すポリゴンを 2 つ作成し、次にそのいずれかがタイプ DOUBLE の 4 つの値で指定された形状領域と交差するかどうかを判別するものです。

```
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
```

結果:

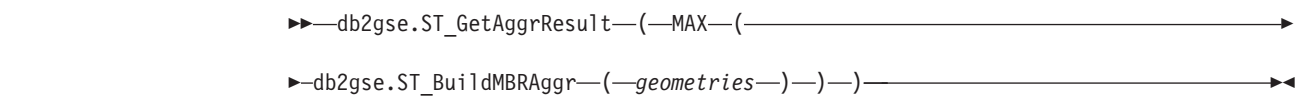
```
Name
--------------------
County_1
County_2
```

MBR 集約

関数 ST_BuildMBRAggr および ST_GetAggrResult の組み合わせは、列の中のすべての形状の最小外接長方形を表す長方形を構成することによって、選択された列の中の形状の列を、単一の形状に集約します。集約を計算するときに、Z 座標と M 座標は破棄されます。

結合するすべての形状が NULL の場合は、NULL が戻されます。形状のすべてが NULL または空である場合は、空の形状が戻されます。結合するすべての形状の最小外接長方形がポイントになった場合は、そのポイントが ST_Point 値として戻されます。結合するすべての形状の最小外接長方形が水平折れ線または垂直折れ線になった場合は、その折れ線が ST_LineString 値として戻されます。それ以外の場合、最小外接長方形が ST_Polygon 値として戻されます。

構文



パラメーター

geometries

ST_Geometry のタイプまたはそのサブタイプの 1 つを持ち、最小の外接長方形を計算するすべての形状を表す、選択された列。

戻りタイプ

db2gse.ST_Geometry

制約事項

以下のいずれかに該当する場合は、全選択内で空間列の和集約を作成できません。

- データベース・パーティション・フィーチャー (DPF) 環境内。
- 全選択で GROUP BY 節を使用した場合。
- DB2 集約関数 MAX 以外の関数を使用した場合。

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次の例は、ST_BuildMBRAggr 関数を使用して、列内のすべての形状の最大外接長方形を得る方法を示しています。この例では、SAMPLE_POINTS 表の GEOMETRY 列にいくつかのポイントが追加されます。この後、SQL コードにより、すべてのポイントの最大外接長方形が決定されます。

```
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
```

結果:

```
Aggregate_of_Points
-----------------------------------------------------------
POLYGON (( 2.00000000 2.00000000, 23.00000000 2.00000000,
23.00000000 15.00000000, 2.00000000 15.00000000, 2.00000000
2.00000000))
```

ST_AppendPoint

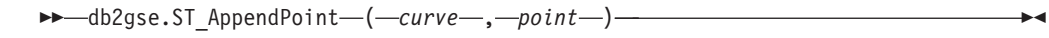
ST_AppendPoint は曲線とポイントを入力パラメーターとし、与えられたポイントにより曲線を拡張します。与えられた曲線が Z または M 座標を持つ場合、ポイントも Z または M 座標を持つ必要があります。結果の曲線は、与えられた曲線の空間参照系で表現されます。

付加されるポイントが曲線と同じ空間参照系で表現されていない場合、他の空間参照系に変換されます。

与えられた曲線が閉じている、または単純な場合には、結果の曲線は閉じていない、または単純でない場合があります。与えられた曲線またはポイントが NULL の場合、または曲線が空の場合は、NULL が戻されます。付加されるポイントが空の場合は、与えられた曲線は変更されずに戻され、警告が出されます (SQLSTATE 01HS3)。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

curve *point* が付加される曲線を表す、ST_Curve タイプまたはそのサブタイプのいずれか 1 つの値。

point *curve* に付加するポイントを表す、ST_Point タイプの値。

戻りタイプ

db2gse.ST_Curve

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

このコードは 2 つの折れ線を作成し、それぞれが 3 つのポイントを持ちます。

```
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) )
```

例 1

この例は、折れ線の終わりにポイント (5, 5) を追加します。

```
SELECT CAST(ST_AsText(ST_AppendPoint(line, ST_Point(5, 5)))
    AS VARCHAR(120)) New
FROM   sample_lines
WHERE  id=1
```

結果:

```
NEW
------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 10.00000000 0.00000000,
0.00000000 0.00000000, 5.00000000 5.00000000)
```

例 2

この例は、Z 座標を持つ折れ線の終わりにポイント (15, 15, 7) を追加します。

```
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
```

結果:

```
NEW
-------------------------------------------------------------
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 5.00000000
5.00000000 5.00000000, 10.00000000 10.00000000 6.00000000,
15.00000000 15.00000000 7.00000000)
```

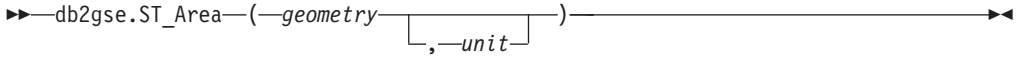
ST_Area

ST_Area は、形状およびオプションとして単位を入力パラメーターとして取り、与えられた形状がカバーするエリアを、デフォルトの、あるいは指定された測定単位で戻します。

形状がポリゴンまたは複数ポリゴンの場合は、その形状によりカバーされるエリアが戻されます。ポイント、折れ線、複数ポイント、および複数折れ線のエリアは 0 (ゼロ) です。形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
そのエリアを判別する形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

unit エリアを測る単位を示す、VARCHAR(128) 値。サポートされる測定単位は DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビューにリストされています。

unit パラメーターを省略すると、次の規則を使用してエリアを測る単位が決められます。

- *geometry* が投影座標システム、または地心から見た座標システムを使用する場合、この座標システムに関連付けられた線形単位が使用されます。
- *geometry* が地理座標システムで、測地座標システム (SRS) でない場合、この座標システムに関連付けられた角度単位が使用されます。
- *geometry* が測地 SRS の場合、デフォルトの測定単位は平方メートルになります。

単位変換の制約事項 : 以下の条件に当てはまる場合には、エラー (SQLSTATE 38SU4) が戻されます。

- 形状の座標システムが指定されておらず、かつ *unit* パラメーターが指定されている。
- 形状の座標システムが投影座標システムで、かつ角度単位が指定されている。
- 形状の座標システムが地理座標システムで、測地 SRS でなく、かつ線形単位が指定されている。
- 形状の座標システムが地理座標システムかつ測地 SRS であり、さらに角度単位が指定されている。

戻りタイプ

DOUBLE

例

例 1

空間分析者は、各販売地域がカバーするエリアのリストを必要としています。販売地域のポリゴンは SAMPLE_POLYGONS 表に保管されています。このエリアは、ST_Area 関数を形状列に適用することにより計算されます。

```
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))
```

次の SELECT ステートメントは、販売地域 ID およびエリアを選択します。

```
SELECT id, ST_Area(geometry) AS area
FROM   sample_polygons
```

結果:

```
ID        AREA
--------- ----------------------
        1   +1.00000000000000E+002
        2   +2.00000000000000E+002
        3   +2.50000000000000E+001
```

例 2

次の SELECT ステートメントは、販売地域 ID およびエリアを各種の単位で選択します。

```
SELECT id,
       ST_Area(geometry) square_feet,
       ST_Area(geometry, 'METER') square_meters,
       ST_Area(geometry, 'STATUTE MILE') square_miles
FROM   sample_polygons
```

結果:

```
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
```

例 3

この例は、State Plane 座標で定義されたポリゴンのエリアを見つけます。

ID が 3 のState Plane 空間参照系は、次のようなコマンドを呼び出すことにより作成されます。

```
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
```

次の SQL ステートメントは、空間参照系 3 内のポリゴンを表に追加し、エリアを平方フィート、平方メートル、および平方マイルで決めます。

```
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;
```

結果:

```
ID   Square Feet               Square Meters             Square Miles
--- ------------------------- ------------------------- -------------------------
  1 +7.94698788000000E+008   +7.38302286101346E+007   +2.85060106320552E+001
```

例 4

空間分析者は、各探査地域がカバーするエリアのリストを必要としています。探査地域ポリゴンは、 SAMPLE_GEODETIC_TAB 表に保存されます。探査地域ポリゴンが対応するのは、以下の地域です。

- 北極を囲む地域
- 南極を囲む地域
- 子午線をまたぐ地域

以下の入力ファイル samp_wkt_rows.txt の 2 番目のフィールドには、これらの地域に対応するポリゴンが含まれています。

```
1|'polygon((5 82,15 82,25 82,35 82,45 82,55 82,65 82,75 82,85 82,95 82,
105 82,115 82,125 82,135 82,145 82,155 82,165 82,175 82,-175 82,-165 82,
-155 82,-145 82,-135 82,-125 82,-115 82,-105 82,-95 82,-85 82,-75 82,
-65 82,-55 82,-45 82,-35 82,-25 82,-15 82,-5 82,5 82))'|'North Pole region'
2|'polygon((175 -82,165 -82,155 -82,145 -82,135 -82,125 -82,115 -82,
105 -82,95 -82,85 -82,75 -82,65 -82,55 -82,45 -82,35 -82,25 -82,15 -82,
5 -82,-5 -82,-15 -82,-25 -82,-35 -82,-45 -82,-55 -82,-65 -82,-75 -82,
-85 -82,-95 -82,-105 -82,-115 -82,-125 -82,-135 -82,-145 -82,-155 -82,
-165 -82,-175 -82,175 -82))'|'South Pole region'
3|'polygon((-175 -42,-175 1,-175 42,175 42,175 -1,175 -42,-175 -42))
'|'180th meridian'
```

以下の SQL ステートメントは、測地参照系 2000000000 のポリゴンを SAMPLE_GEODETIC_TAB 表に追加するものです。

```
SET current function path db2gse;
CREATE TABLE db2se_samp.gsege_temp_samp (
            gid      INTEGER,
            g1_wkt   varchar(500),
            comment  varchar(255)
   ) NOT LOGGED INITIALLY;
LOAD FROM samp_wkt_rows.txt OF DEL MODIFIED BY CHARDEL'' COLDEL|
   INSERT INTO db2se_samp.gsege_temp_samp;

CREATE TABLE sample_geodetic_tab
  (gid INTEGER NOT NULL PRIMARY KEY,
   geometry ST_Geometry),
   comment varchar(255));

INSERT INTO sample_geodetic_tab
     SELECT gid, ST_GeomFromText(g1_wkt, 2000000000), comment
   FROM db2se_samp.gsege_temp_samp;
```

ST_Area 関数は、図系列中のポリゴンのエリアを計算します。 ST_Area のデフォルトの測定単位は平方メートルです。次の SELECT ステートメントは、探査地域 ID およびエリアを平行メートル、平方フィート、平方マイルで選択します。

```
SELECT id, ST_Area(geometry) AS SQUARE_METERS,
ST_Area(geometry,'FOOT') AS SQUARE_FEET,
ST_Area(geometry, 'STATUTE MILE') AS SQUARE_MILES
FROM sample_geodetic_tab
WHERE id BETWEEN 1 AND 9 ORDER BY id;
```

```
ID  SQUARE_METERS            SQUARE_FEET              SQUARE_MILES
--- -----------------------  -----------------------  -----------------------
  1 +2.52472719957839E+012   +2.71759374028922E+013   +9.74802621488040E+005
  2 +2.52475431563494E+012   +2.71762292776957E+013   +9.74813091056005E+005
  3 +9.43568029137069E+012   +1.01564817377028E+014   +3.64313652781464E+006
```

ST_AsBinary

ST_AsBinary は形状を入力パラメーターとして取り、事前割り当てバイナリー表記を戻します。Z 座標と M 座標は破棄され、事前割り当てバイナリー表記で表記されません。

与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
対応する、事前割り当てバイナリー表記に変換される、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

BLOB(2G)

例

次のコードは、ST_AsBinary 関数を使用して、SAMPLE_POINTS 表の形状列にあるポイントを BLOB 列の事前割り当てバイナリー (WKB) 表記に変換する方法を示しています。

```
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))
```

例 1

この例は、GEOMETRY 列から ID 1100 を持つものを、WKB 列に ID 1111 で入れます。

```
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
```

結果:

```
ID          Point
----------- -----------------------------------
       1111 POINT ( 10.00000000 20.00000000)
```

例 2

この例は WKB バイナリー表記を表示します。

```
SELECT id, substr(ST_AsBinary(geometry), 1, 21) AS point_wkb
FROM   sample_points
WHERE  id = 1100
```

結果:

```
ID     POINT_WKB
------ ---------------------------------------------
  1100 x'01010000000000000000002440000000000003440'
```

ST_AsGML

ST_AsGML は形状を入力パラメーターとし、Geography Markup Language (GML) を使用してその形状を表現したものを戻します。

与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

gmlLevel

戻される GML データのフォーマットに使用される GML 仕様レベルを指定するオプション・パラメーター。有効な値は以下のとおりです。

- 2 -<gml:coordinates> タグを使って GML 仕様レベル 2 を使用します。
- 3 -<gml:poslist> タグを使って GML 仕様レベル 3 を使用します。

パラメーターが指定されていない場合、出力は <gml:coord> タグを使って GML 仕様レベル 2 を使用して戻されます。

geometry

対応する GML 表記に変換される、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

CLOB(2G)

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次のコードは、ST_AsGML 関数を使用して GML フラグメントを表示する方法を示しています。この例は、形状列から ID 2222 を持つものを GML 列に入れます。

```
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))
```

次の SELECT ステートメントは、形状の ID および GML 表記をリストします。

```
SELECT id, cast(ST_AsGML(geometry) AS varchar(110)) AS gml_fragment
FROM   sample_points
WHERE  id = 1100
```

結果:

```
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
```

The SELECT statement returns the following result set:

```
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>
```

ST_AsShape

ST_AsShape は形状を入力パラメーターとして使用し、その ESRI 形状表記を戻します。

与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
対応する ESRI 形状表記に変換される、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

BLOB(2G)

例

次のコードは、ST_AsShape 関数を使用して、SAMPLE_POINTS 表の形状列にあるポイントを、形状 BLOB 列の形状バイナリー表記に変換する方法を示しています。この例は、形状列から形状列に入れます。形状のバイナリー表記は、ジオブラウザーで形状を表示する場合 (この場合、形状は ESRI 形状ファイル・フォーマットに準拠する必要がある)、または、形状ファイルの *.SHP ファイル用に形状を作成する場合に使用されます。

```
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
```

結果:

```
ID     SHAPE

------ --------------------------------------------
  1100 x'0100000000000000000024400000000000003440'
```

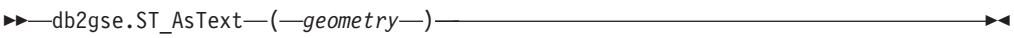
ST_AsText

ST_AsText は、形状を入力パラメーターとして取り、その事前割り当てテキスト表記を戻します。

与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
対応する、事前割り当てテキスト表記に変換される、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

CLOB(2G)

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

データをキャプチャーし、SAMPLE_GEOMETRIES 表に挿入した後、分析者は挿入された値が正しいことを検査するため、形状の事前割り当てテキスト表記を見ます。

```
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))
```

次の SELECT ステートメントは、形状の空間データ・タイプおよび WKT 表記をリストします。形状は ST_AsText 関数によりテキストに変換されます。ST_AsText 関数のデフォルト出力は CLOB(2G) であるため、これはその後 varchar(120) にキャストされます。

```
SELECT id, spatial_type, cast(geometry..ST_AsText
   AS varchar(150)) AS wkt
FROM   sample_geometries
```

結果:

```
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))
```

ST_Boundary

ST_Boundary は形状を入力パラメーターとし、その境界を新しい形状として戻します。結果の形状は、与えられた形状の空間参照系で表現されます。

与えられた形状が、ポイント、複数ポイント、閉じた曲線、または閉じた複数曲線の場合、または与えられた形状が空の場合、結果はタイプ ST_Point の空の形状になります。閉じていない曲線または複数曲線の場合、曲線の開始ポイントと終了ポイントは ST_MultiPoint 値として戻されます (ただし、このポイントが偶数の曲線の開始ポイントまたは終了ポイントでない場合)。面および複数面の場合、与えられた形状の境界を定義する曲線が、ST_Curve または ST_MultiCurve 値として戻されます。与えられた形状が NULL の場合は NULL が戻されます。

可能な場合には、戻される形状のタイプは ST_Point、ST_LineString、または ST_Polygon になります。例えば、穴のないポリゴンの境界は 1 つの折れ線であり、ST_LineString で表されます。1 つまたは複数の穴を持つポリゴンの境界は、ST_MultiLineString で表される複数折れ線からなります。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
ST_Geometry タイプまたはそのサブタイプの 1 つの値。この形状の境界が戻されます。

戻りタイプ

db2gse.ST_Geometry

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、複数の形状を作成し、それぞれの形状の境界を決定します。

```
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

Results
ID      BOUNDARY
------- ------------------------------------------------------------------
      1 LINESTRING ( 40.00000000 120.00000000, 90.00000000 120.00000000,
          90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
          120.00000000)

      2 MULTILINESTRING (( 40.00000000 120.00000000, 90.00000000 120.00000000,
```

```
    90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
    120.00000000),( 70.00000000 130.00000000, 70.00000000 140.00000000,
    80.00000000 140.00000000, 80.00000000 130.00000000, 70.00000000
    130.00000000))

3 MULTIPOINT ( 60.00000000 60.00000000, 70.00000000 70.00000000)

4 MULTIPOINT ( 50.00000000 50.00000000, 70.00000000 70.00000000,
    80.00000000 80.00000000, 90.00000000 90.00000000)

5 POINT EMPTY
```

ST_Buffer

ST_Buffer は、形状、距離、およびオプションとして単位を入力パラメーターとして取り、指定した単位で、指定した距離で与えられた形状を囲む形状を戻します。

結果の形状の境界上の各ポイントは、指定された距離だけ、与えられた形状から離れています。結果の形状は、与えられた形状の空間参照系で表現されます。

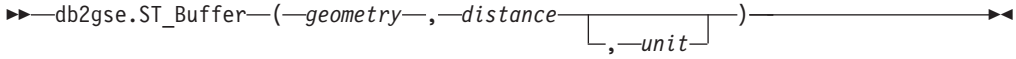
測地データの場合、負の距離を指定すると、ST_Buffer は、入力された形状のすべてのポイントからの距離が、指定された距離 よりも遠い領域を戻します。つまり、負の距離を指定すると、補完的な領域が戻されるということです。

結果の形状の境界内の円弧曲線はすべて、線のストリングによる近似値が求められます。例えば、ポイント周辺のバッファーは円形領域になりますが、これは境界が折れ線のポリゴンで近似値が求められます。

与えられた形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
周りにバッファーを作成する形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。測地データの場合、 ST_Buffer がサポートするデータ・タイプは ST_Point と ST_MultiPoint のみです。

distance
geometry の周りのバッファーに使用される距離を指定する DOUBLE PRECISION 値。測地データの場合、距離を地球の赤道半径より大きくすることはできません。WGS-84 楕円の場合、地球の赤道半径は 6378137.0 m となっています。

unit *distance* を測定する単位を示す VARCHAR(128) 値。サポートされる測定単位は DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビューにリストされています。

unit パラメーターを省略すると、次の規則により distance に使用される測定単位が決められます。

- *geometry* が投影座標システム、または地心から見た座標システムを使用する場合、この座標システムに関連付けられた線形単位がデフォルトになります。
- *geometry* が地理座標システムで、測地座標システム (SRS) でない場合、この座標システムに関連付けられた角度単位がデフォルトになります。
- *geometry* が測地 SRS の場合、デフォルトの測定単位はメートルになります。

単位変換の制約事項 : 以下の条件に当てはまる場合には、エラー (SQLSTATE 38SU4) が戻されます。

- 形状の座標システムが指定されておらず、かつ *unit* パラメーターが指定されている。
- 形状の座標システムが投影座標システムで、かつ角度単位が指定されている。
- 形状の座標システムが地理座標システムで、測地 SRS でなく、かつ線形単位が指定されている。
- 形状の座標システムが地理座標システムかつ測地 SRS であり、さらに角度単位が指定されている。

戻りタイプ

db2gse.ST_Geometry

例

次の例では、結果は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのディスプレイによって異なります。

例 1

次のコードは空間参照系を作成し、SAMPLE_GEOMETRIES 表を作成して、その表にデータを入れます。

```
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))
```

例 2

次の SELECT ステートメントは ST_Buffer 関数を使用して、10 のバッファーを適用します。

```
SELECT id, spatial_type,
       cast(geometry..ST_Buffer(10)..ST_AsText AS varchar(470)) AS buffer_10
FROM   sample_geometries
```

結果:

```
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))
```

例 3

次の SELECT ステートメントは ST_Buffer 関数を使用して、5 のネガティブ・バッファーを適用します。

```
SELECT id, spatial_type,
       cast(ST_AsText(ST_Buffer(geometry, -5)) AS varchar(150))
       AS buffer_negative_5
FROM   sample_geometries
WHERE  id = 3
```

結果:

```
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))
```

例 4

次の SELECT ステートメントは、unit パラメーターを指定してバッファーを適用した結果を示しています。

```
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
```

結果:

```
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))
```

ST_Centroid

ST_Centroid は形状を入力パラメーターとし、形状の中心を戻します。形状の中心とは、与えられた形状の最小外接長方形の中心ポイントです。結果のポイントは、与えられた形状の空間参照系で表現されます。

与えられた形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
その中心を判別する形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

db2gse.ST_Point

例

この例は 2 つの形状を作成し、その図心を見つけます。

```
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
```

結果:

```
ID          CENTROID
----------- ----------------------------------------
          1 POINT ( 65.00000000 135.00000000)
          2 POINT ( 30.00000000 20.00000000)
```

ST_ChangePoint

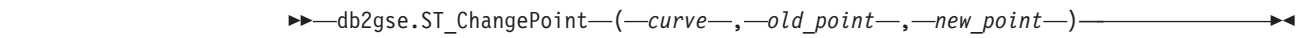
ST_ChangePoint は 1 つの曲線と 2 つのポイントを入力パラメーターとします。これは、与えられた曲線内で、1 番目のポイントと同じポイントをすべて 2 番目のポイントで置き換え、結果の曲線を戻します。結果の形状は、与えられた形状の空間参照系で表現されます。

2 つのポイントが曲線と同じ空間参照系で表現されていない場合、これらのポイントは、曲線に使用されている空間参照系に変換されます。

与えられた曲線が空の場合は、空の値が戻されます。与えられた曲線が NULL 値の場合、または与えられたポイントのいずれかが NULL 値または空の場合は、NULL 値が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

curve *old_point* で示されたポイントが *new_point* に変更される曲線を表す、ST_Curve タイプまたはそのサブタイプの 1 つの値。

old_point
曲線内の、*new_point* に変更されるポイントを示す、ST_Point タイプの値。

new_point
曲線内の、*old_point* で示されたポイントの新しい場所を表す、ST_Point タイプの値。

戻りタイプ

db2gse.ST_Curve

制約事項

曲線内の変更されるポイントは、その曲線の定義に使用されたポイントの 1 つである必要があります。

曲線が Z または M 座標を持つ場合、与えられたポイントも Z または M 座標を持つ必要があります。

例

例 1

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次のコードは SAMPLE_LINES 表を作成し、この表にデータを入れます。

```
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) )
```

例 2

この例は、折れ線内のポイント (5, 5) をすべて、ポイント (6, 6) に変更します。

```
SELECT cast(ST_AsText(ST_ChangePoint(line, ST_Point(5, 5),
                                     ST_Point(6, 6))) as VARCHAR(160))
FROM   sample_lines
WHERE  id=1
```

例 3

この例は、折れ線内のポイント (5, 5, 5) をすべて、ポイント (6, 6, 6) に変更します。

```
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
```

結果:

```
NEW
--------------------------------------------------------------------------------
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 6.00000000 6.00000000
6.00000000, 10.00000000 10.00000000 6.00000000, 5.00000000 5.00000000
7.00000000)
```

ST_Contains

ST_Contains は 2 つの形状を入力パラメーターとし、2 番目の形状が 1 番目の形状に完全に含まれる場合は 1 を戻し、それ以外の場合は 0 (ゼロ) を戻して最初の形状に 2 番目の形状が完全には含まれていないことを示します。

与えられた形状のいずれかが NULL 値または空の場合は、NULL 値が戻されます。

非測地データの場合、2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。測地データの場合は、両方の形状が同じ測地参照系 (SRS) で表現される必要があります。

構文

パラメーター

geometry1

geometry2 を完全に含むかどうかをテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2

geometry1 内に完全に含まれるかどうかをテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

制限事項 : 測地データの場合は、両方の形状が測地で、しかも同じ測地 SRS で表現される必要があります。

戻りタイプ

INTEGER

例

例 1

次のコードは表を作成し、それらの表にデータを入れます。

```
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) )
```

例 2

次のコードは ST_Contains 関数を使用して、どのポイントが特定のポリゴンに含まれるかを判別しています。

```
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
```

結果:

```
POLYGON_ID CONTAINS         POINT_ID
---------- ---------------- --------
       100 does contain            1
       100 does not contain        2
```

例 3

次のコードは ST_Contains 関数を使用して、どの線が特定のポリゴンに含まれるかを判別しています。

```
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
```

結果:

```
POLYGON_ID CONTAINS         LINE_ID
---------- ---------------- -------
       100 does contain          10
       100 does not contain      20
```

ST_ConvexHull

ST_ConvexHull は形状を入力パラメーターとし、その凸包を戻します。

結果の形状は、与えられた形状の空間参照系で表現されます。

可能な場合には、戻される形状のタイプは ST_Point、ST_LineString、または ST_Polygon になります。例えば、穴のないポリゴンの境界は 1 つの折れ線であり、ST_LineString で表されます。1 つまたは複数の穴を持つポリゴンの境界は、ST_MultiLineString で表される複数折れ線からなります。

与えられた形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
凸包を計算する形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

db2gse.ST_Geometry

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次のコードは SAMPLE_GEOMETRIES 表を作成し、この表にデータを入れます。

```
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))
```

次の SELECT ステートメントは、上で作成されたすべての形状の凸包を計算し、結果を表示します。

```
SELECT id, spatial_type, cast(geometry..ST_ConvexHull..ST_AsText
   AS varchar(300)) AS convexhull
FROM   sample_geometries
```

結果:

```
ID     SPATIAL_TYPE       CONVEXHULL
------ ------------------ --------------------------------------------------
    1 ST_LineString      POLYGON (( 20.00000000 40.00000000,
                           20.00000000 20.00000000, 30.00000000
                           30.00000000, 30.00000000 50.00000000,
                           20.00000000 40.00000000))

    2 ST_Polygon         POLYGON (( 110.00000000 140.00000000,
                           110.00000000 120.00000000, 120.00000000
                           130.00000000,  110.00000000 140.00000000))

    3 ST_Polygon         POLYGON (( 15.00000000 50.00000000,
                           25.00000000 35.00000000, 30.00000000
                           30.00000000, 60.00000000 30.00000000,
                           75.00000000 40.00000000, 80.00000000
                           90.00000000, 40.00000000 85.00000000,
                           35.00000000 80.00000000, 15.00000000
                           50.00000000))

    4 ST_MultiPoint      POLYGON (( 20.00000000 40.00000000,
                           20.00000000 20.00000000, 30.00000000
                           30.00000000, 30.00000000 50.00000000,
                           20.00000000 40.00000000))
```

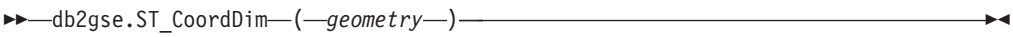
ST_CoordDim

ST_CoordDim は形状を入力パラメーターとし、その座標のディメンション数を戻します。

与えられた形状が Z および M 座標を持たない場合、ディメンション数は 2 です。Z 座標があり M 座標がない場合、または M 座標があり Z 座標がない場合、ディメンション数は 3 です。Z 座標と M 座標があればディメンション数は 4 です。形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
ディメンション数を検索しようとする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

この例は、複数の形状を作成し、それらの座標のディメンション数を決定します。

```
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
```

結果:

```
ID              COORDDIM
--------------- -----------
Empty Point               2
Linestring                2
Polygon                   2
Multipoint M              3
Multipoint Z              3
Point ZM                  4
```

ST_Crosses

ST_Crosses は 2 つの形状を入力パラメーターとし、1 番目の形状が 2 番目と交わる場合に 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。

1 番目の形状がポリゴンまたは複数ポリゴンの場合、または 2 番目の形状がポイントまたは複数ポイントの場合、あるいは形状のいずれかが NULL 値または空である場合は、NULL が戻されます。2 つの形状の交差が、指定された 2 つの形状の最大ディメンションより 1 少ないディメンションを持つ形状になり、結果の形状が指定された 2 つの形状のどちらとも等しくない場合は、1 が戻されます。それ以外の場合、結果は 0 (ゼロ) です。

構文

パラメーター

geometry1
geometry2 との交わりをテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2
geometry1 と交わっているかどうかをテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

次のコードは、作成された形状がお互いに交わるかどうかを判別します。

```
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
```

結果:

```
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
```

ST_Difference

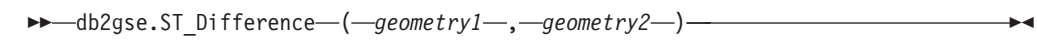
ST_Difference は、2 つの形状を入力パラメーターとし、1 番目の形状が 2 番目と交わらない部分を戻します。

2 つの形状は、同じディメンションの形状でなければなりません。いずれかの形状が NULL の場合は NULL が戻されます。 1 番目の形状が空の場合、タイプ ST_Point の空の形状が戻されます。2 番目の形状が空の場合は、1 番目の形状が変更されずに戻されます。

非測地データの場合、2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。測地データの場合は、両方の形状が同じ測地参照系 (SRS) で表現される必要があります。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry1
差を *geometry2* に計算するために使用する、1 番目の形状を表す、ST_Geometry タイプの値。

geometry2
geometry1 との差を計算するために使用する、2 番目の形状を表す、ST_Geometry タイプの値。

測地データの制約事項

- 両方の形状が測地で、しかも同じ測地 SRS で表現される必要があります。
- ST_Difference がサポートするデータ・タイプは、ST_Point、ST_Polygon、ST_MultiPoint、 ST_MultiPolygon のみです。

戻りタイプ

db2gse.ST_Geometry

戻される形状のディメンションは、入力された形状と同じになります。

例

次の例では、結果は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのディスプレイによって異なります。

次のコードは SAMPLE_GEOMETRIES 表を作成し、この表にデータを入れます。

```
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))
```

例 1

この例は 2 つの分離したポリゴンの差を見つけます。

```
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
```

結果:

```
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))
```

例 2

この例は 2 つの交差するポリゴンの差を見つけます。

```
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
```

結果:

```
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))
```

例 3

この例は 2 つの重なり合う折れ線の差を見つけます。

```
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
```

結果:

```
ID       ID       DIFFERENCE
-------- -------- --------------------------------------------------
       4        5 LINESTRING ( 70.00000000 70.00000000, 75.00000000
                    75.00000000)
```

ST_Dimension

ST_Dimension は形状を入力パラメーターとし、そのディメンションを戻します。

与えられた形状が空の場合は -1 が戻されます。ポイントおよび複数ポイントの場合のディメンションは 0 (ゼロ)、曲線および複数曲線の場合のディメンションは 1、ポリゴンおよび複数ポリゴンの場合のディメンションは 2 です。与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
そのディメンションを戻す形状を表す、ST_Geometry タイプの値。

戻りタイプ

INTEGER

例

この例は、複数の異なる形状を作成し、それらのディメンションを見つけます。

```
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
```

結果:

```
ID              DIMENSION
--------------- -----------
Empty Point              -1
Point ZM                  0
MultiPoint M              0
LineString                1
Polygon                   2
```

ST_Disjoint

ST_Disjoint は 2 つの形状を入力パラメーターとし、与えられた形状が交差しない場合は 1 を戻します。形状が交差する場合は、0 (ゼロ) が戻されます。

2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。

与えられた形状のいずれかが NULL または空の場合は、NULL 値が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry1
geometry2 との交差をテストする形状を表す、ST_Geometry タイプの値。

geometry2
geometry1 との交差をテストする形状を表す、ST_Geometry タイプの値。

戻りタイプ

INTEGER

例

例 1

次のコードは SAMPLE_GEOMETRIES 表に複数の形状を作成します。

```
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))
```

例 2

この例は、最初のポリゴンがどの形状とも交わっていないかどうかを判別します。

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 1
```

結果:

```
ID          ID          DISJOINT
----------- ----------- -----------
          1           1           0
          1           2           0
          1           3           1
          1           4           1
          1           5           0
```

例 3

この例は、3 番目のポリゴンがどの形状とも交わっていないかどうかを判別します。

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 3
```

結果:

```
ID          ID          DISJOINT
----------- ----------- -----------
          3           1           1
          3           2           0
          3           3           0
          3           4           0
          3           5           0
```

例 4

この例は、2 番目の折れ線がどの形状とも交わっていないかどうかを判別します。

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 5
```

結果:

```
ID          ID          DISJOINT
----------- ----------- -----------
          5           1           0
          5           2           0
          5           3           0
          5           4           1
          5           5           0
```

ST_Distance

ST_Distance は、2 つの形状およびオプションとして単位を入力パラメーターとして取り、1 番目の形状内の任意のポイントと 2 番目の形状内の任意のポイントとの間の最短距離を、デフォルトの、あるいは指定された単位で戻します。

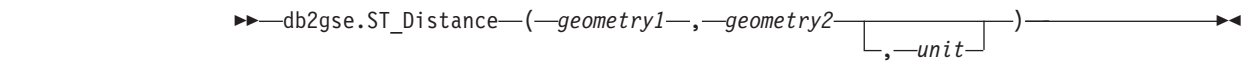
測地データの場合、ST_Distance は、任意の 2 つの形状間の*測地距離* を戻します。測地距離とは、楕円表面での最短距離のことです。

2 つの形状のいずれかが NULL または空の場合は、NULL が戻されます。

非測地データの場合、2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。測地データの場合は、両方の形状が同じ測地参照系 (SRS) で表現される必要があります。

計測単位を指定すると、この関数をメソッドとして呼び出すこともできます。

構文



パラメーター

geometry1
geometry2 との距離を計算するために使用する形状を表す、ST_Geometry タイプの値。

geometry2
geometry1 との距離を計算するために使用する形状を表す、ST_Geometry タイプの値。

unit 結果を測定する単位を示す、VARCHAR(128) 値。サポートされる測定単位は DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビューにリストされています。

測地データの場合は、両方の形状が同じ測地 SRS で表現される必要があります。

unit パラメーターを省略すると、次の規則により、結果に使用される測定単位が決められます。

- *geometry1* が投影座標システム、または地心から見た座標システムを使用する場合、この座標システムに関連付けられた線形単位がデフォルトになります。
- *geometry1* が地理座標システムで、測地座標システム (SRS) でない場合、この座標システムに関連付けられた角度単位がデフォルトになります。

- *geometry1* が測地 SRS の場合、デフォルトの測定単位は m (メートル) になります。

単位変換の制約事項 : 以下の条件に当てはまる場合には、エラー (SQLSTATE 38SU4) が戻されます。

- 形状の座標システムが指定されておらず、かつ *unit* パラメーターが指定されている。
- 形状の座標システムが投影座標システムで、かつ角度単位が指定されている。
- 形状の座標システムが地理座標システムかつ測地 SRS であり、さらに角度単位が指定されている。

戻りタイプ

DOUBLE

例

例 1

次の SQL ステートメントは、SAMPLE_GEOMETRIES1 表と SAMPLE_GEOMETRIES2 表を作成し、そこにデータを入れます。

```
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) )
```

例 2

次の SELECT ステートメントは、SAMPLE_GEOMTRIES1 表と SAMPLE_GEOMTRIES2 表にある各種の形状間の距離を計算します。

```
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
```

結果:

```
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
```

例 3

次の SELECT ステートメントは、お互いが距離 100 以内にあるすべての形状の見つけ方を示しています。

```
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
```

結果:

```
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
```

例 4

次の SELECT ステートメントは、各種の形状間の距離をキロメートルで計算します。

```
SAMPLE_GEOMTRIES1 and SAMPLE_GEOMTRIES2 tables.
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
```

結果:

```
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
```

ST_DistanceToPoint

ST_DistanceToPoint は、入力パラメーターとして単一曲線または複数曲線の形状と 1 つのポイント形状を取り、指定したポイントまでの曲線形状に沿った距離を返します。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

curve-geometry
処理する形状を表す、ST_Curve タイプまたは ST_MultiCurve タイプまたはそのサブタイプの 1 つの値。

point-geometry
指定した曲線に沿ったポイントであるタイプ ST_Point の値。

戻りタイプ

DOUBLE

例

例 1

以下の SQL ステートメントにより、2 つの列を持つ SAMPLE_GEOMETRIES 表が作成されます。ID 列で各行が一意的に識別されます。 GEOMETRY ST_LineString 列にはサンプル形状が格納されます。

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

以下の SQL ステートメントにより、2 つの行が SAMPLE_GEOMETRIES 表に挿入されます。

```
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))
```

以下の SELECT ステートメントおよび対応する結果セットは、ST_DistanceToPoint 関数を使用して位置 (1.5, 15.0) にあるポイントまでの距離を検出する方法を示しています。

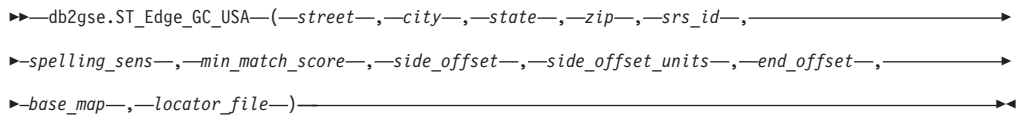
```
SELECT ID, DECIMAL(ST_DistanceToPoint(geometry,ST_Point(1.5,15.0,1)),10,5)
AS DISTANCE FROM sample_geometries

ID          DISTANCE
----------- ------------
          1     15.07481
          2     85.42394

  2 record(s) selected.
```

ST_Edge_GC_USA

ST_Edge_GC_USA は、アメリカ合衆国に存在するアドレスをポイントにジオコーディングする DB2SE_USA_GEOCODER をインプリメントする関数です。アドレスは EDGE ファイルと比較されます。EDGE ファイルはダウンロード可能なジオコーダー参照データで提供されます。

この関数は、ストリート番号と名前、市の名前、州、郵便番号、および結果のポイントの空間参照系 ID を入力パラメーターとして取り、ST_Point 値を戻します。また、ジオコーディング処理に影響するいくつかの構成パラメーターを指定できます。

構文



パラメーター

street ジオコーディングされるアドレスのストリート番号と名前を含む値 (タイプは VARCHAR(128))。

この値は NULL にはできません。

city ジオコーディングされるアドレスの市の名前を含む値 (タイプは VARCHAR(128))。

zip パラメーターを指定した場合は、この値を NULL にすることができます。

state ジオコーディングされるアドレスの州の名前を含む値 (タイプは VARCHAR(128))。州は省略形にすることも、略さずに指定することもできます。

zip パラメーターを指定した場合は、この値を NULL にすることができます。

zip ジオコーディングされるアドレスの郵便番号を含む値 (タイプは VARCHAR(10))。郵便番号は 5 桁で、または 5+4 表記で指定できます。

city および *state* パラメーターを指定した場合は、この値を NULL にすることができます。

srs_id 結果のポイントの空間参照系の数値 ID を含む値 (タイプは INTEGER)。この値は、地理座標システム GCS_NORTH_AMERICAN_1983 に基づいて投影座標システムを使用する、既存の空間参照系を示すか、またはこの地理座標システム自体 (GCS_NORTH_AMERICAN_1983) を使用する既存の空間参照系を示す必要があります。

srs_id がカタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、エラーが戻されます (SQLSTATE 38SU1)。

spelling_sens
与えられたアドレスに適用すべきスペリングの感度を指定する値 (タイプは INTEGER)。この値は 0 (ゼロ) から 100 の範囲内でなければなりません。この値が大きいほど、与えられたアドレスのスペリングの相違に関してジオコーダーはより厳格になります。スペリングに相違があると、突き合わせの最終スコアに適用されるペナルティーがより高くなります。

スペリングの感度をあまりに高く設定すると、ジオコーディングに成功するアドレスは少なくなり、代わりに NULL が戻されます。スペリングの感度を低く設定しすぎると、アドレスのスペリングの相違を受け入れるレベルが低くなるため、一致しないアドレスでも正しく一致したと見なされる可能性が高くなります。**推奨事項:** この値を 60 に設定します。

この値が NULL の場合、スペリングの感度はロケーター・ファイルから取られます。ロケーター・ファイルにこれが指定されていない場合は、スペリングの感度として 60 が使用されます。

min_match_score
与えられたアドレスと一致したと見なされるポイントがもつべき、最小のスコア値を含む値 (タイプは INTEGER)。最小スコア値は 0 (ゼロ) から 100 の範囲内である必要があります。ポイントのスコアが *min_match_score* 値より低い場合は、ポイントではなく NULL が戻され、アドレスはジオコーディングされません。

アドレスがポイントのスコアに影響する要因には、基本マップの品質、スペリングの感度、あるいは正確性など、いろいろあります。**推奨事項:** この値を 80 に設定します。

この値が NULL の場合、最小一致スコアはロケーター・ファイルから取られます。ロケーター・ファイルにこれが指定されていない場合は、最小スコア値として 80 が使用されます。

side_offset
結果のポイントをストリートの中心からどれだけ離して置くかを指定する値 (タイプは DOUBLE)。この値は 0 (ゼロ) 以上である必要があります。*side_offset_unit* パラメーターは、サイド・オフセットを測定するために使用する単位を示します。

この値が NULL の場合、サイド・オフセットはロケーター・ファイルから取られます。ロケーター・ファイルにこれが指定されていない場合は、サイド・オフセットとして 0.0 が使用されます。

side_offset_units

side_offset パラメーターを測定する単位を含む値 (タイプは VARCHAR(128))。値は次の単位のいずれかである必要があります。

- Inches
- Points
- Feet
- Yards
- Miles
- Nautical miles
- Millimeters
- Centimeters
- Meters
- Kilometers
- 度 (10進数)
- Projected meters
- Reference data units

この値が NULL の場合、サイド・オフセット単位はロケーター・ファイルから取られます。ロケーター・ファイルにこれが指定されていない場合は、サイド・オフセットは feet (フィート) で測定されます。

end_offset

ストリート・セグメントのちょうど終わりにあるポイントを、セグメント内にどのくらい離して置くかを示す値 (タイプは INTEGER)。この値は 0 (ゼロ) 以上である必要があります。このパラメーターは、結果のポイントをストリートの中央の交点に置くことを避けるために使用されます。終了オフセットは、基本マップ上のポイント (可能な最小の精度) で測定されます。

この値が NULL の場合、終了オフセットはロケーター・ファイルから取られます。ロケーター・ファイルにこれが指定されていない場合は、終了オフセットとして 3 が使用されます。

base_map

基本マップ (.edg) ファイルを指す、基本名を含む完全修飾パスを含む値 (タイプは VARCHAR(256))。基本マップ・ファイルは、与えられたアドレスを突き合わせるためにジオコーダーが使用します。DB2 Spatial Extender が提供する基本マップを使用する必要があります。このパラメーターは、基本マップを別のディレクトリーに置いていれば使用できます。

この値が NULL の場合、基本マップへのパスはロケーター・ファイルから取られます。ロケーター・ファイルにこれが指定されていない場合は、gse/refdata サブディレクトリー内の、現行インスタンスの sqllib ディレクトリーで、基本マップを探します。探すファイルの基本名は usa.edg です。

locator_file

ジオコーダーの追加の構成パラメーターを含むロケーター・ファイルを指す、基本名を含む完全修飾パスが入った値 (タイプは VARCHAR(256))。DB2 Spatial Extender が提供するロケーター・ファイルを使用する必要があります。

この値が NULL の場合、gse/cfg/geocoder サブディレクトリー内の、現行インスタンスの sqllib ディレクトリーで、ロケーター・ファイルを探します。探すファイルの基本名は EDGELocator.loc です。

戻りタイプ

db2gse.ST_Point

例

例 1

次のコードは、表 SAMPLE_GEOCODING を作成し、2 つのアドレスを挿入します。このアドレスを後でジオコーディングします。与えられたアドレスの最小一致スコアは 50 に設定し、結果のポイントの空間参照系は 1 です。

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geocoding (
 street VARCHAR(128),
  city   VARCHAR(128),
 state  VARCHAR(128),
 zip    VARCHAR(5) )

INSERT INTO geocoding(street, city, state, zip)
 VALUES ('1212 New York Ave NW', 'Washington', 'DC', '20005'),
 ('100 First North Street', 'San Jose', 'CA', NULL)

SELECT VARCHAR(ST_AsText(ST_Edge_GC_USA(street, city, state, zip, 1,
    CAST(NULL AS INTEGER), 50, CAST(NULL AS DOUBLE),
    CAST(NULL AS VARCHAR(128)), CAST(NULL AS INTEGER),
     CAST(NULL AS VARCHAR(256)), CAST(NULL AS VARCHAR(256)))), 50)
FROM sample_geocoding
```

結果:

```
1
--------------------------------------------------
POINT ( -77.02829300 38.90049000)
POINT ( -121.94507200 37.28766700)
```

例 2

この例では、投影座標システムを使用する空間参照系を作成します。ジオコーディング関数のインターフェースを単純化するため、ユーザー定義関数を作成し、ST_Edge_GC_USA 関数をそこに含めます。

```
db2se create_srs <db_name> -srsName CALIFORNIA -srsId 101 -xScale 1
  -coordsysName NAD_1983_STATEPLANE_CALIFORNIA_I_FIPS_0401

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE FUNCTION California_GC (
  street VARCHAR(128), city VARCHAR(128), zip VARCHAR(10))
 RETURNS db2gse.ST_Point
   LANGUAGE SQL
   RETURN db2gse.ST_Edge_GC_USA(street, city, 'CA', zip, 101,
     CAST(NULL AS INTEGER), CAST(NULL AS INTEGER),
     CAST(NULL AS DOUBLE), CAST(NULL AS VARCHAR(128)),
       CAST(NULL AS INTEGER), CAST(NULL AS VARCHAR(256)))

CREATE TABLE sample_geocoding (
  street VARCHAR(128),
```

```
  city  VARCHAR(128),
  state  VARCHAR(128),
    zip    VARCHAR(5) )

INSERT INTO geocoding(street, city, state, zip)
VALUES ('100 First North Street', 'San Jose', 'CA', NULL)

SELECT VARCHAR(ST_AsText(California_GC(street, city, zip)), 50)
FROM sample_geocoding
```

結果:

```
1
--------------------------------------------------
POINT ( 2004879.00000000 272723.00000000)
```

ポイントの X 座標と Y 座標の値が最初の例と異なりますが、その理由は異なる空間参照系を使用しているためです。

ST_Endpoint

ST_Endpoint は曲線を入力パラメーターとし、その曲線の最後のポイントを戻します。結果のポイントは、与えられた曲線の空間参照系で表現されます。

与えられた曲線が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

curve　最後のポイントを戻す形状を表す、ST_Curve タイプの値。

戻りタイプ

db2gse.ST_Point

例

SELECT ステートメントは、SAMPLE_LINES 表内のそれぞれの形状の終了ポイントを見つけます。

```
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
```

結果:

```
ID          ENDPOINT
----------- --------------------------------------------------
          1 POINT ( 0.00000000 10.00000000)
          2 POINT Z ( 5.00000000 5.00000000 7.00000000)
```

ST_Envelope

ST_Envelope は形状を入力パラメーターとし、形状の周りのエンベロープを戻します。エンベロープはポリゴンとして表現される長方形です。

与えられた形状がポイント、水平線、または垂直線の場合、与えられた形状よりも少し大きい長方形が戻されます。それ以外の場合、最小外接長方形がエンベロープとして戻されます。与えられた形状が NULL または空の場合は、NULL が戻されます。すべての形状の正確な最小外接長方形を戻すには、関数 ST_MBR を使用してください。

測地データの場合、エンベロープは、形状の最小の外接円を囲むポリゴンになります。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
エンベロープを戻す形状を表す、ST_Geometry タイプの値。

戻りタイプ

db2gse.ST_Polygon

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、複数の形状を作成し、次にそれらのエンベロープを決定します。空でないポイントおよび折れ線 (水平) の場合、エンベロープは形状よりも少し大きい長方形です。

```
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
```

結果:

```
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))
```

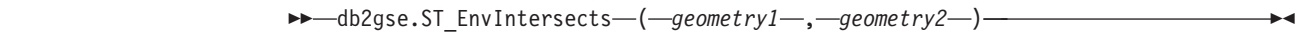
ST_EnvIntersects

ST_EnvIntersects は 2 つの形状を入力パラメーターとし、2 つの形状のエンベロープが交差する場合は 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。

いずれかの形状が NULL または空の場合は、NULL 値が戻されます。

構文



パラメーター

geometry1

geometry2 のエンベロープとの交差をテストするエンベロープの、形状を表す ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2

geometry1 のエンベロープとの交差をテストするエンベロープの、形状を表す ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

この例は 2 つの並行した折れ線を作成し、それらの交点をチェックします。折れ線自体は交差しませんが、そのエンベロープは交差します。

```
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
```

結果:

```
ID          ID          INTERSECTS  ENVELOPE_INTERSECTS
----------- ----------- ----------- -------------------
          1           2           0                   1
```

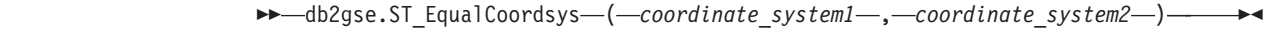
ST_EqualCoordsys

ST_EqualCoordsys は 2 つの座標システム定義を入力パラメーターとし、与えられた定義が同一の場合は整数値 1 を戻します。それ以外の場合、整数値 0 (ゼロ) が戻されます。

座標システム定義は、スペース、括弧、大文字小文字、および浮動小数点表記の相違に関係なく、比較されます。

与えられた座標システム定義のいずれかが NULL の場合は、NULL が戻されます。

構文



パラメーター

coordinate_system1
coordinate_system2 と比較される 1 番目の座標システムを定義する VARCHAR(2048) タイプの値。

coordinate_system2
coordinate_system1 と比較される 2 番目の座標システムを定義する VARCHAR(2048) タイプの値。

戻りタイプ

INTEGER

例

この例は 2 つのオーストラリア座標システムを比較し、この 2 つが同じかどうかを調べます。

```
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')
      )
```

結果:

```
1
-----------
          0
```

ST_Equals

ST_Equals は 2 つの形状を入力パラメーターとし、形状が等しい場合は 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。形状の定義に使用されるポイントの順序は、同一性のテストに関係しません。

2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。

与えられた 2 つの形状のいずれかが NULL の場合は、NULL が戻されます。

構文

パラメーター

geometry1
geometry2 と比較される形状を表す、ST_Geometry タイプの値。

geometry2
geometry1 と比較される形状を表す、ST_Geometry タイプの値。

戻りタイプ

INTEGER

例

例 1

この例は、異なる順序で座標システムを持つ 2 つのポリゴンを作成します。これらのポリゴンが等しいと見なされることを示すため、ST_Equal を使用します。

```
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
```

結果:

```
ID          ID          EQUALS
----------- ----------- -----------
          1           2           1
```

例 2

この例では、X 座標と Y 座標が同じで、M 座標 (指標) が異なる 2 つの形状を作成します。ST_Equal 関数を使用して形状を比較すると、0 (ゼロ) が戻され、これらの形状が等しくないことが示されます。

```
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
```

結果:

```
ID          ID          EQUALS
----------- ----------- -----------
          3           4           0
```

例 3

この例では、異なる座標のセットを使用して 2 つの形状を作成していますが、両方とも同じ形状を表します。ST_Equal はこれらの形状を比較し、両方の形状がまったく等しいことを示します。

```
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
```

結果:

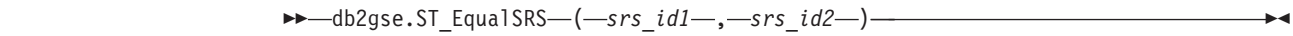
```
ID          ID          EQUALS
----------- ----------- -----------
          5           6           1
```

ST_EqualSRS

ST_EqualSRS は 2 つの空間参照系 ID を入力パラメーターとし、与えられた空間参照系が同一の場合は 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。オフセット、スケール係数、および座標システムが比較されます。

与えられた空間参照系 ID のいずれかが NULL の場合は、NULL が戻されます。

構文



パラメーター

srs_id1

srs_id2 で示された空間参照系と比較される、最初の空間参照系を識別する INTEGER タイプの値。

srs_id2

srs_id1 で示された空間参照系と比較される、2 番目の空間参照系を識別する INTEGER タイプの値。

戻りタイプ

INTEGER

例

db2se を次のように呼び出して、2 つの類似の空間参照系を作成します。

```
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
```

これらの SRS は同じオフセットとスケール値を持ち、同じ座標システムを参照します。唯一の違いは、定義された名前と SRS ID です。したがって、比較は 1 を戻し、これらが同じであることを示します。

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

VALUES ST_EqualSRS(12, 22)
```

結果:

```
1
-----------
          1
```

ST_ExteriorRing

ST_ExteriorRing はポリゴンを入力パラメーターとし、その外部リングを曲線として戻します。結果の曲線は、与えられたポリゴンの空間参照系で表現されます。

与えられたポリゴンが NULL または空の場合は、NULL が戻されます。ポリゴンに内部リングがない場合、戻される外部リングはポリゴンの境界と同じです。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

polygon
外部リングを戻すポリゴンを表す、ST_Polygon タイプの値。

戻りタイプ

db2gse.ST_Curve

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は 2 つのポリゴン (1 つは 2 つの内部リングを持ち、もう 1 つは内部リングを持たない) を作成し、次にその外部リングを決定します。

```
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
```

結果:

```
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)
```

ST_FindMeasure または ST_LocateAlong

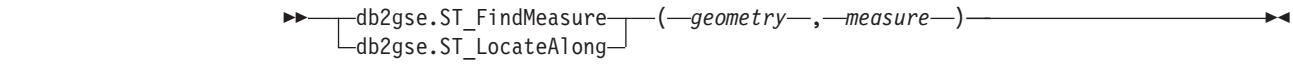
ST_FindMeasure または ST_LocateAlong は形状と指標を入力パラメーターとし、指定された指標を含んでいる、指定された形状の指定された指標そのものをもつ形状部分の、複数ポイントまたは複数曲線を戻します。

ポイントおよび複数ポイントについては、指定された指標をもっているすべてのポイントが戻されます。曲線、複数曲線、面、および複数面の場合、結果を計算するために補間が行われます。面および複数面の計算は、形状の境界に関して行われます。

ポイントおよび複数ポイントの場合、与えられた指標が見つからない場合は、空の形状が戻されます。他のすべての形状については、与えられた指標が形状内の最小の指標より低い場合、または形状内の最高の指標より高い場合は、空の形状が戻されます。与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
M 座標 (指標) に *measure* が含まれる部分を検索する形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

measure
geometry の一部が結果に含まれていなければならない指標を示す、DOUBLE タイプの値。

戻りタイプ

db2gse.ST_Geometry

例

例 1

次の CREATE TABLE ステートメントは、SAMPLE_GEOMETRIES 表を作成します。SAMPLE_GEOMETRIES には、ID 列 (各行を一意的に識別する列)、および GEOMETRY ST_Geometry 列 (サンプルの形状を保管する列) の 2 つの列があります。

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id SMALLINT, geometry ST_GEOMETRY)
```

次の INSERT ステートメントは 2 つの行を挿入します。最初の行は折れ線であり、2 番目の行は複数ポイントです。

```
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))
```

例 2

次の SELECT ステートメントおよび対応する結果セットでは、ST_FindMeasure 関数を使用して指標が 7 のポイントを見つけています。最初の行は 1 つのポイントを戻します。しかし、2 番目の行は空のポイントを戻します。線形フィーチャー (0 より大きいディメンションを持つ形状) の場合、ST_FindMeasure はポイントを補間できますが、複数ポイントの場合、ターゲットの指標は正確に一致する必要があります。

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 7))
   AS varchar(45)) AS measure_7
FROM   sample_geometries
```

結果:

```
ID     MEASURE_7
------ ---------------------------------------------
     1 POINT M ( 3.50000000 3.50000000 7.00000000)
     2 POINT EMPTY
```

例 3

次の SELECT ステートメントおよび対応する結果セットで、ST_FindMeasure 関数は 1 つのポイントと 1 つの複数ポイントを戻します。ターゲットの指標 6 は、ST_FindMeasure および複数ポイントのソース・データの両方の指標と一致します。

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 6))
   AS varchar(120)) AS measure_6
FROM   sample_geometries
```

結果:

```
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)
```

ST_Generalize

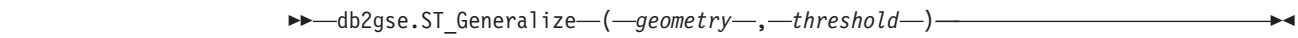
ST_Generalize は形状としきい値を入力パラメーターとし、形状の一般的特性を保持しつつ、ポイントの数を減らして、与えられた形状を表現します。

Douglas-Peucker line-simplification (ダグラス・デッカーの線単純化) アルゴリズムを使用し、これにより、ポイントの並びを直線セグメントで置き換えることができるようになるまで、形状を定義する一連のポイントを繰り返し分割します。この線セグメント内では、定義されたポイントはすべて、与えられたしきい値を超えて直線セグメントから離れることはありません。Z および M 座標は単純化には考慮されません。結果の形状は、与えられた形状の空間参照系にあります。

指定された形状が空の場合、タイプ ST_Point の空の形状が戻されます。指定された形状またはしきい値が NULL の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
線の単純化を適用する形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

threshold
線単純化アルゴリズムに使用するしきい値を示す、タイプ DOUBLE の値。しきい値は 0 (ゼロ) 以上である必要があります。しきい値を大きくするほど、一般化された形状を表現するために使用されるポイントの数は少なくなります。測地データの場合、しきい値の単位はメートルです。

戻りタイプ

db2gse.ST_Geometry

例

次の例では、結果は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのディスプレイによって異なります。

例 1

(10, 10) から (80, 80) に行く 8 つのポイントを持つ折れ線を作成します。パスはほとんど直線ですが、ポイントのいくつかは線から少し離れています。
ST_Generalize 関数を使用して、線上のポイントの数を減らすことができます。

```
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))
```

例 2

一般化係数として 3 を使用すると、折れ線は 4 つの座標に減らされるが、元の折れ線の表現に非常に近いものです。

```
SELECT CAST(ST_AsText(ST_Generalize(geometry, 3)) as VARCHAR(115))
  Generalize_3
FROM sample_lines
```

結果:

```
GENERALIZE 3
------------------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 34.00000000 26.00000000,
  59.00000000  63.00000000, 80.00000000 80.00000000)
```

例 3

一般化係数として 6 を使用すると、折れ線はたった 2 つの座標に減らされます。これは上の例よりも単純な折れ線になりますが、元の表現からはより大きく乖離 (かいり) することになります。

```
SELECT CAST(ST_AsText(ST_Generalize(geometry, 6)) as VARCHAR(65))
  Generalize_6
FROM sample_lines
```

結果:

```
GENERALIZE 6
------------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 80.00000000 80.00000000)
```

ST_GeomCollection

形状の集合を作成するには ST_GeomCollection を使用します。

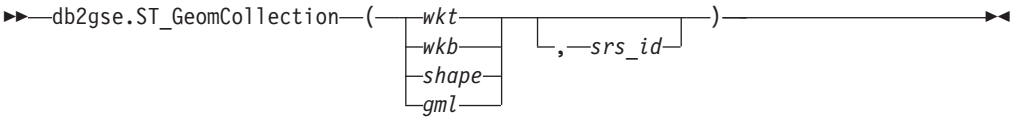
ST_GeomCollection は、次の入力の 1 つから形状の集合を作成します。

- 事前割り当てテキスト表記
- 事前割り当てバイナリー表記
- ESRI 形状表記
- Geography Markup Language (GML) 表記

結果の形状集合を入れる空間参照系を示すため、オプションの空間参照系 ID を指定することができます。

事前割り当てテキスト表記、事前割り当てバイナリー表記、ESRI 形状表記、または GML 表記が NULL の場合は、NULL が戻されます。

構文



パラメーター

wkt 結果の形状集合の事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。

wkb 結果の形状集合の事前割り当てバイナリー表記が入る、タイプ BLOB(2G) の値。

shape 結果の形状集合の ESRI 形状表記を表す、BLOB(2G) タイプの値。

gml Geography Markup Language (GML) を使用した、結果の形状集合を表す、CLOB(2G) タイプの値。

srs_id 結果の形状集合の空間参照系を識別する、タイプ INTEGER の値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が暗黙的に使用されます。

srs_id がカタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、エラーが戻されます (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_GeomCollection

注

srs_id パラメーターを省略すると、*wkt* および *gml* を明示的に CLOB タイプにキャストする必要がある場合があります。さもなければ、 DB2 は参照タイプ REF(ST_GeomCollection) から ST_GeomCollection タイプに値をキャストするために使用される関数にゆだねる可能性があります。次の例では、必ず DB2 は正しい関数に解決をゆだねます。

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次のコードは、ST_GeomCollection 関数を使用して、事前割り当てテキスト (WKT) 表記から複数ポイント、複数線、および複数ポリゴンを、また Geography Markup Language (GML) から複数ポイントを作成し、GeomCollection 列に挿入する方法を示しています。

```
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
```

結果:

```
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)
```

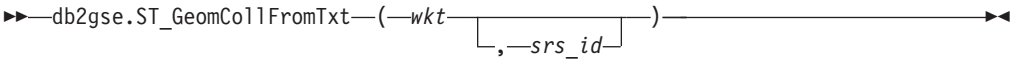
ST_GeomCollFromTxt

ST_GeomCollFromTxt は、形状集合の事前割り当てテキスト表記および、オプションとして空間参照系 ID を入力パラメーターとして取り、対応する形状集合を戻します。

指定された事前割り当てテキスト表記が NULL の場合は、NULL が戻されます。

同じ結果を得るには ST_GeomCollection 関数をお勧めします。お勧めする理由は、ST_GeomCollection は事前割り当てバイナリー表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkt 結果の形状集合の事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。

srs_id 結果の形状集合の空間参照系を識別する、タイプ INTEGER の値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が暗黙的に使用されます。

srs_id がカタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、エラーが戻されます (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_GeomCollection

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次のコードは、ST_GeomCollFromTxt 関数を使用して、事前割り当てテキスト (WKT) 表記から複数ポイント、複数線、および複数ポリゴンを作成し、GeomCollection 列に挿入する方法を示しています。

```
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
```

結果:

```
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)))
```

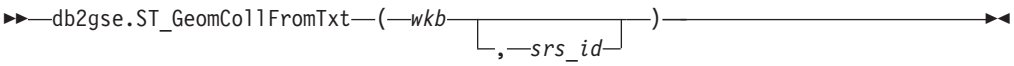
ST_GeomCollFromWKB

ST_GeomCollFromWKB は、入力パラメーターとして、形状集合の事前割り当てバイナリー表記および、オプションの空間参照系 ID を取り、対応する形状集合を戻します。

与えられた、事前割り当てバイナリー表記が NULL の場合は、NULL が戻されます。

この機能は、ST_GeomCollection バージョンを使用することをお勧めします。

構文



パラメーター

wkb 結果の形状集合の事前割り当てバイナリー表記が入る、タイプ BLOB(2G) の値。

srs_id 結果の形状集合の空間参照系を識別する、タイプ INTEGER の値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が暗黙的に使用されます。

srs_id がカタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、エラーが戻されます (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_GeomCollection

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次のコードは ST_GeomCollFromWKB 関数を使用して、事前割り当てバイナリー表記の形状集合の座標を作成し、照会する方法を示しています。ID 4021 と ID 4022 を持ち、空間参照系 1 の形状集合を持つ行を SAMPLE_GEOMCOLLECTION 表に挿入しています。

```
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
```

結果:

```
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))
```

ST_Geometry

ST_Geometry は、次の入力の 1 つから形状を作成します。

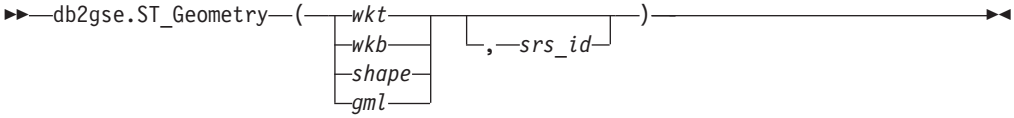
- 事前割り当てテキスト表記
- 事前割り当てバイナリー表記
- ESRI 形状表記
- Geography Markup Language (GML) 表記

結果の形状を入れる空間参照系を示すため、オプションとして空間参照系 ID を指定することができます。

結果の形状の動的タイプは、ST_Geometry のインスタンス化可能なサブタイプの 1 つです。

事前割り当てテキスト表記、事前割り当てバイナリー表記、ESRI 形状表記、または GML 表記が NULL の場合は、NULL が戻されます。

構文



パラメーター

wkt 結果の形状の事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。

wkb 結果の形状の事前割り当てバイナリー表記が入る、タイプ BLOB(2G) の値。

shape 結果の形状の ESRI 形状表記を表す、BLOB(2G) タイプの値。

gml Geography Markup Language (GML) を使用した、結果の形状を表す、CLOB(2G) タイプの値。

srs_id 結果の形状の空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が暗黙的に使用されます。

srs_id がカタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、エラーが戻されます (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_Geometry

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次のコードは、ST_Geometry 関数を使用して、事前割り当てテキスト (WKT) ポイント表記からポイントを、または Geographic Markup Language (GML) の線表記から線を、作成し、挿入する方法を示しています。

ST_Geometry 関数は、各種の形状表記からどのような空間データ・タイプでも作成できるので、空間データ・タイプ作成機能の中で最も柔軟性があります。ST_LineFromText は WKT 線表記から線を作成できるだけです。ST_WKTToSql はどのようなタイプでも作成できますが、WKT 表記からのみです。

```
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
```

結果:

```
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)
```

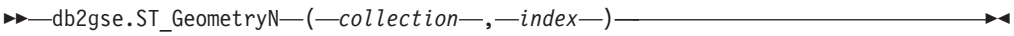
ST_GeometryN

ST_GeometryN は、形状の集合と 1 つの索引を入力パラメーターとし、集合の中から、索引で指定された形状を戻します。結果の形状は、与えられた形状集合の空間参照系で表現されます。

与えられた形状集合が NULL または空の場合、または索引が 1 より小さいか集合内の形状の数より大きい場合は NULL が戻され、警告条件が起こります (01HS0)。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

collection

その中にある *n* 番目の形状を検索する形状集合を表す、ST_GeomCollection タイプまたはそのサブタイプの 1 つの値。

index *collection* から戻される *n* 番目の形状を示す、INTEGER タイプの値。

index が 1 より小さいか集合内の形状の数より大きい場合は、NULL および警告 (SQLSTATE 01HS0) が戻されます。

戻りタイプ

db2gse.ST_Geometry

例

次のコードは、形状集合の中の 2 番目の形状を選択する方法を示しています。

```
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
```

結果:

```
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))
```

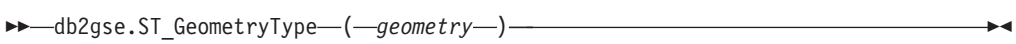
ST_GeometryType

ST_GeometryType は形状を入力パラメーターとし、その形状の動的タイプの完全修飾されたタイプ名を戻します。

DB2 関数 TYPE_SCHEMA と TYPE_NAME は同じ効果を持ちます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
形状タイプを戻す、タイプ ST_Geometry の値。

戻りタイプ

VARCHAR(128)

例

次のコードは、形状のタイプを判別する方法を示しています。

```
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
```

結果:

```
ID          GEOMETRY_TYPE
----------- ------------------------------
       7101 "DB2GSE  "."ST_POINT"
       7102 "DB2GSE  "."ST_LINESTRING"
       7103 "DB2GSE  "."ST_POLYGON"
       7104 "DB2GSE  "."ST_MULTIPOINT"
```

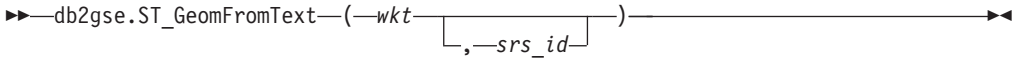
ST_GeomFromText

ST_GeomFromText は、形状の事前割り当てテキスト表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応する形状を戻します。

指定された事前割り当てテキスト表記が NULL の場合は、NULL が戻されます。

この機能としては、ST_Geometry の使用をお勧めします。

構文



パラメーター

wkt 結果の形状の事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。

srs_id 結果の形状の空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が暗黙的に使用されます。

srs_id がカタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、エラーが戻されます (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_Geometry

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例では、ST_GeomFromText 関数を使用して、事前定義されたテキスト (WKT) のポイント表記からポイントを作成し、挿入しています。

次のコードは、ID および、WKT 表記を使用する空間参照系 1 の形状を持つ行を SAMPLE_POINTS 表に挿入します。

```
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))
```

次の SELECT ステートメントは、SAMPLE_GEOMETRIES 表から ID と GEOMETRIES を戻します。

```
SELECT id, cast(geometry..ST_AsText AS varchar(105))
   AS geometry
FROM   sample_geometries
```

結果:

```
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))
```

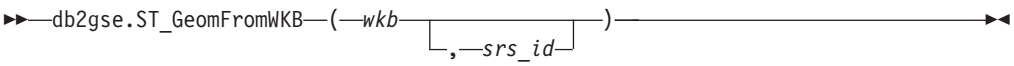
ST_GeomFromWKB

ST_GeomFromWKB は、形状の事前割り当てバイナリー表記および、オプションとして空間参照系 ID を入力パラメーターとして取り、対応する形状を戻します。

与えられた、事前割り当てバイナリー表記が NULL の場合は、NULL が戻されます。

この機能としては、ST_Geometry の使用をお勧めします。

構文



パラメーター

wkb 結果の形状の事前割り当てバイナリー表記が入る、タイプ BLOB(2G) の値。

srs_id 結果の形状の空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が暗黙的に使用されます。

指定された srs_id パラメーターが、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系を示していない場合は、エラーが戻されます (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_Geometry

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次のコードは ST_GeomFromWKB 関数を使用して、事前割り当てバイナリー (WKB) 線表記から線を作成し、挿入する方法を示しています。

次の例は、ID および空間参照系 1 の WKB 表記の形状を持つレコードを SAMPLE_GEOMETRIES 表に挿入しています。

```
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
```

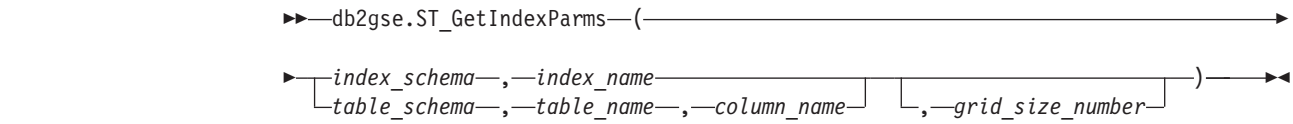
結果:

```
 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))
```

ST_GetIndexParms

ST_GetIndexParms は、空間インデックスの ID または空間列の ID のいずれかを入力パラメーターとし、索引または空間列の索引を定義するために使用されるパラメーターを戻します。追加のパラメーター番号を指定すると、その番号で示されたグリッド・サイズだけが戻されます。

構文



パラメーター

index_schema

修飾されていない名前 *index_name* を持つ空間インデックスが属するスキーマを示す、VARCHAR(128) タイプの値。スキーマ名は大文字小文字の区別があり、SYSCAT.SCHEMATA カタログ・ビューにリストされている必要があります。

このパラメーターが NULL の場合、空間インデックスのスキーマ名として、CURRENT SCHEMA 特殊レジスターの値が使用されます。

index_name

索引パラメーターを戻させたい空間インデックスの、修飾されていない名前が入る、VARCHAR(128) タイプの値。索引名は大文字小文字の区別があり、スキーマ *index_schema* 用に SYSCAT.INDEXES カタログ・ビューにリストされている必要があります。

table_schema

修飾されていない名前 *table_name* を持つ表が属するスキーマを示す、VARCHAR(128) タイプの値。スキーマ名は大文字小文字の区別があり、SYSCAT.SCHEMATA カタログ・ビューにリストされている必要があります。

このパラメーターが NULL の場合、空間インデックスのスキーマ名として、CURRENT SCHEMA 特殊レジスターの値が使用されます。

table_name

空間列 *column_name* を持つ表の、修飾されていない名前を含む、VARCHAR(128) タイプの値。表名は大文字小文字の区別があり、スキーマ *table_schema* 用に SYSCAT.TABLES カタログ・ビューにリストされている必要があります。

column_name

列の空間インデックスの索引パラメーターが戻される、表 *table_schema.table_name* の列を示す、VARCHAR(128) タイプの値。列名は大文字小文字の区別があり、表 *table_schema.table_name* 用に SYSCAT.COLUMNS カタログ・ビューにリストされている必要があります。

列に空間インデックスが定義されていない場合は、エラーが起こります (SQLSTATE 38SQ0)。

grid_size_number

値を戻させたいパラメーターを識別する、DOUBLE タイプの値。

この値が 1 より小さいか、または 3 より大きい場合は、エラーが戻されます (SQLSTATE 38SQ1)。

戻りタイプ

DOUBLE (*grid_size_number* が指定されている場合)

grid_size_number が指定されていない場合は、2 つの列 ORDINAL および VALUE を持つ表が戻されます。列 ORDINAL のタイプは INTEGER、列 VALUE のタイプは DOUBLE です。

グリッド索引のパラメーターが戻される場合、ORDINAL 列には、1 番目のグリッド・サイズの値 1、2 番目の値 2、3 番目の値 3 が入ります。列 VALUE にはグリッド・サイズが入ります。

VALUE 列には、それぞれのパラメーターに該当する値が入ります。

例

例 1

次のコードは、空間列と空間インデックスを持つ表を作成します。

```
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)
```

ST_GetIndexParms 関数を使用して、空間インデックスの作成時に使用したパラメーターの値を検索することができます。

例 2

この例は、どのパラメーターを戻すかを明示的に番号で指定して、空間グリッド索引の 3 つのグリッド・サイズを別々に検索する方法を示しています。

```
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 1)
```

結果:

```
1
------------------------
  +1.00000000000000E+000
```

```
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 2)
```

結果:

```
1
------------------------
  +1.00000000000000E+001
```

```
VALUES ST_GetIndexParms('SCH', 'IDX', 3)
```

結果:

```
1
------------------------
  +1.00000000000000E+003
```

例 3

この例は、空間グリッド索引のすべてのパラメーターの検索方法を示しています。ST_GetIndexParms 関数は、パラメーター番号および対応するグリッド・サイズを示す表を戻します。

```
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION') ) AS t
```

結果:

```
ORDINAL     VALUE
----------- ------------------------
          1   +1.00000000000000E+000
          2   +1.00000000000000E+001
          3   +1.00000000000000E+003
```

```
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'IDX') ) AS t
```

結果:

```
ORDINAL     VALUE
----------- ------------------------
          1   +1.00000000000000E+000
          2   +1.00000000000000E+001
          3   +1.00000000000000E+003
```

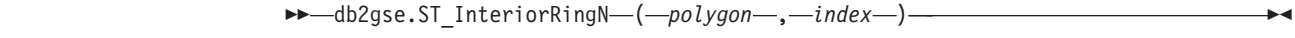
ST_InteriorRingN

ST_InteriorRingN は、ポリゴンと索引を入力パラメーターとし、与えられた索引で指定された内部リングを折れ線として戻します。内部リングは、内部形状検査ルーチンにより定義された規則に従って編成されます。

与えられたポリゴンが NULL または空の場合、またはその中に内部リングがない場合は、NULL が戻されます。索引が 1 より小さいかポリゴン内の内部リングの数より大きい場合は、NULL 値が戻され、警告条件が起こります (1HS1)。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

polygon
index で指定された内部リングを戻す形状を表す、ST_Polygon タイプの値。

index 戻される *n* 番目 の内部リングを示す INTEGER タイプの値。*index* で識別される内部リングがない場合は、警告条件が発生します (01HS1)。

戻りタイプ

db2gse.ST_Curve

例

この例では、2 つの内部リングを持つポリゴンを作成します。次に ST_InteriorRingN 呼び出しを使用して、2 番目の内部リングを検索します。

```
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
```

結果:

```
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)
```

ST_Intersection

ST_Intersection は 2 つの形状を入力パラメーターとし、与えられた 2 つの形状の交差を表す形状を戻します。交差は、1 番目の形状の 2 番目の形状と重なり合う部分です。結果の形状は、1 番目の形状の空間参照系で表現されます。

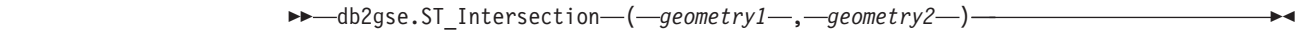
可能な場合には、戻される形状のタイプは ST_Point、ST_LineString、または ST_Polygon になります。例えば、 点とポリゴンの交差は、空白か、1 つの点になり、ST_MultiPoint で表されます。

2 つの形状のいずれかが NULL の場合は、NULL が戻されます。

非測地データの場合、2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。測地データの場合は、両方の形状が同じ測地参照系 (SRS) で表現される必要があります。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry1

geometry2 との交差を計算する、1 番目の形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2

geometry1 との交差を計算する、2 番目の形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

測地データの場合は、両方の形状が同じ測地 SRS で表現される必要があります。

戻りタイプ

db2gse.ST_Geometry

戻される形状のディメンションは、測地データの折れ線を除き、入力された形状のうちディメンションの低い方に合わされます。測地データの場合、2 つの折れ線の交差のディメンションは 0 になります (これは、交差がポイント、あるいは複数ポイントになるということ) 。

例

次の例では、結果は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのディスプレイによって異なります。

この例は、複数の異なる形状を作成し、次に最初の形状との交差 (もしあれば) を判別しています。

```
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
```

結果:

```
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 record(s) selected.
```

ST_Intersects

ST_Intersects は 2 つの形状を入力パラメーターとし、与えられた形状が交差する場合は 1 を戻します。形状が交差しない場合は、0 (ゼロ) が戻されます。

2 つの形状のいずれかが NULL または空の場合は、NULL が戻されます。

非測地データの場合、2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。測地データの場合は、両方の形状が同じ測地参照系 (SRS) で表現される必要があります。

構文

パラメーター

geometry1

geometry2 との交差をテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2

geometry1 との交差をテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

制限事項: 測地データの場合は、両方の形状が測地で、しかも同じ測地 SRS で表現される必要があります。

戻りタイプ

INTEGER

例

次のステートメントは、SAMPLE_GEOMETRIES1 表と SAMPLE_GEOMETRIES2 表を作成し、そこにデータを入れます。

```
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) )
```

次の SELECT ステートメントは、SAMPLE_GEOMTRIES1 表と SAMPLE_GEOMTRIES2 表にある各種の形状が交差するかどうかを判別します。

```
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
```

結果:

```
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
```

ST_Is3d

ST_Is3d は形状を入力パラメーターとし、与えられた形状が Z 座標を持つ場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

与えられた形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
Z 座標の存在をテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

この例では、Z 座標と M 座標 (指標) を持つ形状、または持たない形状をいくつか作成しています。次に ST_Is3d を使用して、どの形状に Z 座標が含まれているかを判別しています。

```
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
```

結果:

```
ID          IS_3D
----------- -----------
          1           0
          2           0
```

```
3           0
4           1
5           1
```

ST_IsClosed

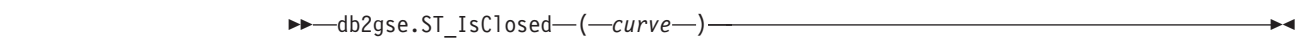
ST_IsClosed は曲線または複数曲線を入力パラメーターとし、与えられた曲線または複数曲線が閉じている場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

曲線は、開始ポイントと終了ポイントが等しい場合、閉じています。曲線が Z 座標を持つ場合、Z 座標の開始ポイントと終了ポイントは等しくなければなりません。そうでない場合、ポイントは等しいと見なされず、曲線は閉じていません。複数曲線は、曲線のそれぞれが単体で閉じている場合に、閉じています。

与えられた曲線または複数曲線が空の場合は、0 (ゼロ) が戻されます。これが NULL の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

curve テストする曲線または複数曲線を表す、ST_Curve タイプまたは ST_MultiCurve タイプ、あるいはこれらのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

例 1

この例は、複数の折れ線を作成します。最後の 2 つの折れ線は同じ X 座標と Y 座標を持ちますが、1 つの折れ線は変化する Z 座標を持つため折れ線は閉じておらず、もう 1 つの折れ線は変化する M 座標 (指標) を持ちますが、これは折れ線が閉じているかどうかに影響しません。

```
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
```

結果:

```
ID          IS_CLOSED
----------- -----------
          1           0
          2           0
          3           1
          4           1
          5           0
```

例 2

この例は、2 つの複数折れ線を作成します。複数折れ線が閉じているかどうかを判別するため、ST_IsClosed を使用します。最初の複数折れ線は、すべての曲線を一緒にすれば完全に閉じたループになりますが、閉じていません。これは、それぞれの曲線自体は閉じていないからです。

2 番目の複数折れ線は、それぞれの曲線自体が閉じているので、閉じています。

```
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
```

結果:

```
ID          IS_CLOSED
----------- -----------
          6           0
          7           1
```

ST_IsEmpty

ST_IsEmpty は形状を入力パラメーターとし、与えられた形状が空の場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
テストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

次のコードは 3 つの形状を作成し、次にそれらが空であるかどうかを判別します。

```
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
```

結果:

```
ID          IS_EMPTY
----------- -----------
          1           1
          2           0
          3           0
          4           0
          5           0
```

ST_IsMeasured

ST_IsMeasured は形状を入力パラメーターとし、与えられた形状が M 座標 (指標) を持つ場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

与えられた形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
M 座標 (指標) の存在をテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

この例では、Z 座標と M 座標 (指標) を持つ形状、または持たない形状をいくつか作成しています。次に ST_IsMeasured を使用して、どの形状に指標が含まれているかを判別します。

```
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
```

結果:

```
ID          IS_MEASURED
----------- -----------
          1           0
          2           0
          3           1
          4           0
          5           1
```

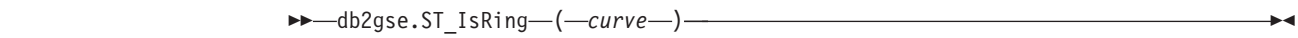
ST_IsRing

ST_IsRing は曲線を入力パラメーターとし、これがリングであれば 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。曲線は、単純かつ閉じていればリングです。

与えられた曲線が空の場合は、0 (ゼロ) が戻されます。これが NULL の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

curve テストする曲線を表す、ST_Curve タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

この例では、4 つの折れ線を作成します。これらがリングであるかをチェックするために、ST_IsRing を使用します。最後の折れ線は閉じていますが、パスがそれ自体を横切っているため、リングとは見なされません。

```
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
```

結果:

```
ID          IS_CLOSED   IS_RING
----------- ----------- -----------
          1           1           0
          2           0           0
          3           1           1
          4           1           0
```

ST_IsSimple

ST_IsSimple は形状を入力パラメーターとし、与えられた形状が単純な場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

ポイント、面、および複数面は常に単純です。曲線は、同じポイントを 2 回通らなければ単純です。複数ポイントは、2 つのポイントが同一でなければ単純です。複数曲線は、その曲線のすべてが単純であり、かつ、複数曲線内の曲線の境界上にあるポイントでのみ交差が起こる場合に、単純です。

与えられた形状が空の場合は 1 が戻されます。これが NULL の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
テストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

この例では、いくつかの形状を作成し、それらが単純であるかどうかをチェックします。ID 4 の形状は、同一のポイントを複数持つので、単純とは見なされません。ID 6 の形状は、折れ線がそれ自体を横切っているので、単純とは見なされません。

```
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
```

結果:

```
ID          IS_SIMPLE
----------- -----------
          1           1
          2           1
          3           1
          4           0
          5           1
          6           0
          7           1
```

ST_IsValid

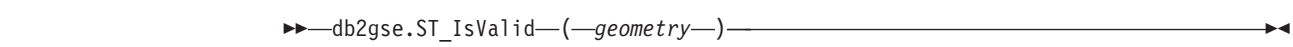
ST_IsValid は形状を入力パラメーターとし、それが有効な場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

形状は、構造化されたタイプのすべての属性が形状データの内部表記と整合していて、その内部表記が壊れていない場合にのみ、有効です。

与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

この例は複数の形状を作成し、ST_IsValid を使用してそれらの形状が有効であるかチェックします。ST_Geometry のようなコンストラクター・ルーチンは、無効な形状の作成を許さないので、形状はすべて有効です。

```
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
```

結果:

```
ID          IS_VALID
----------- -----------
          1           1
          2           1
          3           1
          4           1
          5           1
```

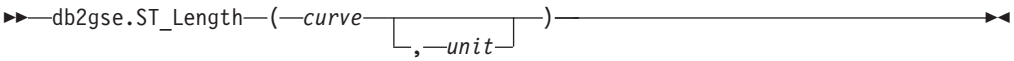
ST_Length

ST_Length は、曲線または複数曲線および (オプションとして) 単位を入力パラメーターとし、与えられた曲線または複数曲線の長さを、デフォルトの、あるいは与えられた測定単位で戻します。

与えられた曲線または複数曲線が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

curve 長さを戻す曲線を表す、ST_Curve または ST_MultiCurve タイプの値。

unit 曲線の長さを測る単位を示す、VARCHAR(128) の値。サポートされる測定単位は DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビューにリストされています。

unit パラメーターを省略すると、次の規則を使用して長さを測る単位が決められます。

- *curve* が投影座標システム、または地心から見た座標システムを使用する場合、この座標システムに関連付けられた線形単位がデフォルトになります。
- *curve* が地理座標システムで、測地座標システム (SRS) でない場合、この座標システムに関連付けられた角度単位がデフォルトになります。
- *curve* が測地 SRS の場合、デフォルトの測定単位はメートルになります。

単位変換の制約事項 : 以下の条件に当てはまる場合には、エラー (SQLSTATE 38SU4) が戻されます。

- *curve* の座標システムが指定されておらず、かつ *unit* パラメーターが指定されている。
- *curve* の座標システムが投影座標システムで、かつ角度単位が指定されている。

- *curve* が地理座標システムで、測地 SRS でなく、かつ線形単位が指定されている。
- *curve* 測地 SRS で、かつ角度単位が指定されている。

戻りタイプ

DOUBLE

例

例 1

次の SQL ステートメントは、表 SAMPLE_GEOMETRIES を作成し、その表に線および複数線を挿入します。

```
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))
```

例 2

次の SELECT ステートメントは、SAMPLE_GEOMTRIES 表にある線の長さを計算します。

```
SELECT id, spatial_type, cast(ST_Length(geometry..ST_ToLineString)
   AS DECIMAL(7, 2)) AS "Line Length"
FROM   sample_geometries
WHERE  id = 1110
```

結果:

```
ID     SPATIAL_TYPE         Line Length
------ -------------------- -----------
  1110 ST_LineString              10.00
```

例 3

次の SELECT ステートメントは、SAMPLE_GEOMTRIES 表にある複数線の長さを計算します。

```
SELECT id, spatial_type, ST_Length(ST_ToMultiLine(geometry))
   AS multiline_length
FROM   sample_geometries
WHERE  id = 1111
```

結果:

```
ID     SPATIAL_TYPE         MULTILINE_LENGTH
 ------ -------------------- ------------------------
   1111 ST_MultiLineString     +2.76437123387202E+001
```

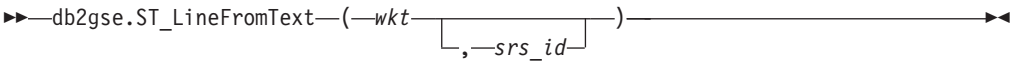
ST_LineFromText

ST_LineFromText は、折れ線の事前割り当てテキスト表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応する折れ線を戻します。

指定された事前割り当てテキスト表記が NULL の場合は、NULL が戻されます。

この機能としては、ST_LineString をお勧めします。

構文



パラメーター

wkt 結果の折れ線の事前割り当てテキスト表記が入る、CLOB(2G) タイプの値。

srs_id 結果の折れ線の空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

srs_id がカタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、エラーが戻されます (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_LineString

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次のコードは ST_LineFromText 関数を使用して、事前割り当てテキスト (WKT) 線表記から線を作成し、挿入しています。SAMPLE_LINES 表に、ID および、空間参照系 1 で WKT 表記の線値を持つ行が挿入されます。

```
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
```

結果:

```
ID     LINESTRING
------ ---------------------------------------------------------------
  1110 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)
  1111 LINESTRING EMPTY
```

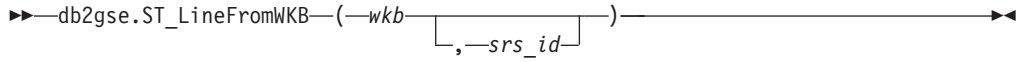
ST_LineFromWKB

ST_LineFromWKB は、折れ線の事前割り当てバイナリー表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応する折れ線を戻します。

与えられた、事前割り当てバイナリー表記が NULL の場合は、NULL が戻されます。

この機能としては、ST_LineString をお勧めします。

構文



パラメーター

wkb 結果の折れ線の事前割り当てバイナリー表記が入る、BLOB(2G) タイプの値。

srs_id 結果の折れ線の空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

srs_id がカタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、エラーが戻されます (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_LineString

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次のコードは ST_LineFromWKB 関数を使用して、事前割り当てバイナリー表記から線を作成し、挿入しています。SAMPLE_LINES 表に、ID および、空間参照系 1 で WKB 表記の線を持つ行を挿入しています。

```
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
```

結果:

```
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)
```

ST_LineString

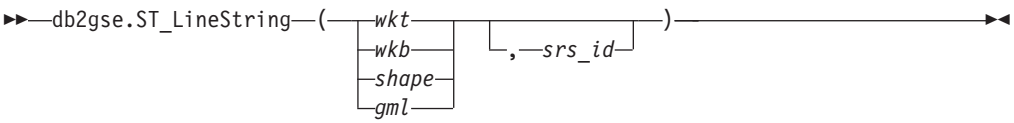
ST_LineString は、次の入力の 1 つから折れ線を作成します。

- 事前割り当てテキスト表記
- 事前割り当てバイナリー表記
- ESRI 形状表記
- Geography Markup Language (GML) 表記

結果の折れ線を置く空間参照系を示すため、オプションとして空間参照系 ID を指定することができます。

事前割り当てテキスト表記、事前割り当てバイナリー表記、ESRI 形状表記、または GML 表記が NULL の場合は、NULL が戻されます。

構文



パラメーター

wkt 結果のポリゴンの事前割り当てテキスト表記が入る、CLOB(2G) タイプの値。

wkb 結果のポリゴンの事前割り当てバイナリー表記が入る、BLOB(2G) タイプの値。

shape 結果のポリゴンの ESRI 形状表記を表す、BLOB(2G) タイプの値。

gml Geography Markup Language (GML) を使用した、結果のポリゴンを表す、CLOB(2G) タイプの値。

srs_id 結果のポリゴンの空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

srs_id がカタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、エラーが戻されます (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_LineString

例

次のコードは ST_LineString 関数を使用して、事前割り当てテキスト (WKT) 線表記または事前割り当てバイナリー (WKB) 表記から線を作成し、挿入します。

次の例は、ID および空間参照系 1 で WKT と GML 表記の線を持つ行を、SAMPLE_LINES 表に挿入します。

```
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
```

結果:

```
ID     LINESTRING
------ ---------------------------------------------------------------
  1110 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)
  1111 LINESTRING ( 90.00000000 90.00000000, 100.00000000 100.00000000)
```

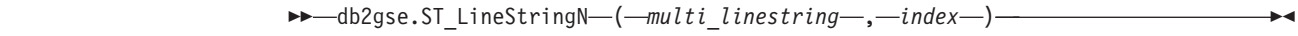
ST_LineStringN

ST_LineStringN は、複数折れ線と索引を入力パラメーターとし、その索引で識別される折れ線を戻します。結果の折れ線は、与えられた複数折れ線の空間参照系で表現されます。

与えられた複数折れ線が NULL または空の場合、または索引が 1 より小さいか折れ線の数より大きい場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

multi_linestring
index で識別される折れ線を戻す複数折れ線を表す、ST_MultiLineString タイプの値。

index *multi_linestring* から戻される *n* 番目の折れ線を示す、INTEGER タイプの値。

index が 1 より小さいか *multi_linestring* 内の折れ線の数より大きい場合は、NULL および警告条件 (SQLSTATE 01HS0) が戻されます。

戻りタイプ

db2gse.ST_LineString

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

次の SELECT ステートメントは、SAMPLE_MLINES 表内の複数折れ線の中にある 2 番目の形状を選択する方法を示しています。

```
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
```

結果:

```
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)
```

ST_M

ST_M は次のことを行います。

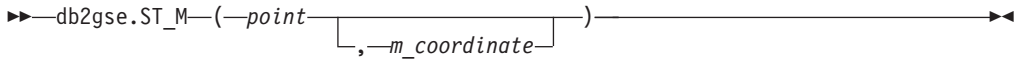
- ポイントを入力パラメーターとし、その M (指標) 座標を戻します。
- ポイントと M 座標を入力パラメーターとして取り、指定されたポイントに M 座標が存在しない場合でも、ポイント自体を、与えられた指標にセットされた M 座標と一緒に戻します。

指定された M 座標が NULL の場合、そのポイントの M 座標は除去されます。

指定されたポイントが NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

point M 座標を戻すか変更したい値 (タイプは ST_Point)。

m_coordinate

point の新規 M 座標を表す DOUBLE タイプの値。

m_coordinate が NULL の場合、M 座標は *point* から除去されます。

戻りタイプ

- *m_coordinate* が指定されていない場合は、DOUBLE
- *m_coordinate* が指定されている場合は、db2gse.ST_Point

例

例 1

この例は、ST_M 関数の使用法を示しています。3 つのポイントを作成し、SAMPLE_POINTS 表に挿入します。これらはすべて、ID が 1 の空間参照系にあります。

```
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))
```

例 2

この例は、SAMPLE_POINTS 表内のポイントの M 座標を見つけます。

```
SELECT id, ST_M (geometry) M_COORD
  FROM sample_points
```

結果:

```
ID          M_COORD
----------- ------------------------
          1 +5.00000000000000E+000
          2 +4.00000000000000E+000
          3 +7.00000000000000E+000
```

例 3

この例は、M 座標が 40 にセットされているポイントの 1 つを戻します。

```
SELECT id, CAST (ST_AsText (ST_M (geometry, 40) )
  AS VARCHAR(60) ) M_COORD_40
  FROM sample_points
  WHERE id=3
```

結果:

```
ID         M_COORD_40
---------- ----------------------------------------------------
         3 POINT ZM (3.00000000 8.00000000 23.00000000 40.00000000)
```

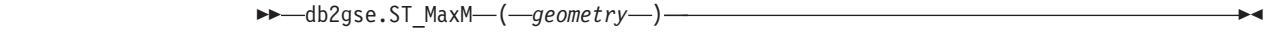
ST_MaxM

ST_MaxM は形状を入力パラメーターとし、その最大 M 座標を戻します。

与えられた形状が NULL または空の場合、または M 座標がない場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
最大 M 座標を戻す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

DOUBLE

例

例 1

この例は、ST_MaxM 関数の使用法を示しています。3 つのポリゴンを作成し、SAMPLE_POLYS 表に挿入します。

```
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) )
```

例 2

この例は、SAMPLE_POLYS 内の各ポリゴンの最大の M 座標を見つけます。

```
SELECT id, CAST ( ST_MaxM(geometry) AS INTEGER) MAX_M
  FROM sample_polys
```

結果:

```
ID          MAX_M
----------- ------------
          1            4
          2           12
          3           16
```

例 3

この例は、GEOMETRY 列のすべてのポリゴンに対して存在する最大の M 座標を見つけます。

```
SELECT CAST ( MAX ( ST_MaxM(geometry) ) AS INTEGER) OVERALL_MAX_M
  FROM sample_polys
```

結果:

```
OVERALL_MAX_M
--------------
           16
```

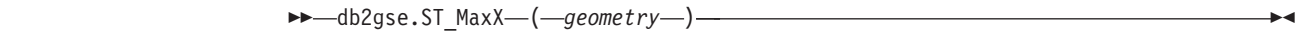
ST_MaxX

ST_MaxX は形状を入力パラメーターとし、その最大 X 座標を戻します。

与えられた形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
最大 X 座標を戻す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

DOUBLE

例

例 1

この例は、ST_MaxX 関数の使用法を示しています。3 つのポリゴンを作成し、SAMPLE_POLYS 表に挿入します。3 番目の例は、最大と最小の座標値を戻すすべての関数を使用して、特定の空間列に保管された形状の地理範囲を算定する方法を示しています。

```
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) )
```

例 2

この例は、SAMPLE_POLYS 内の各ポリゴンの最大の X 座標を見つけます。

```
SELECT id, CAST ( ST_MaxX(geometry) AS INTEGER) MAX_X_COORD
  FROM sample_polys
```

結果:

```
ID          MAX_X_COORD
----------- ------------
          1          120
          2            5
          3           12
```

例 3

この例は、GEOMETRY 列のすべてのポリゴンに対して存在する最大の X 座標を見つけます。

```
SELECT CAST ( MAX ( ST_MaxX(geometry) ) AS INTEGER) OVERALL_MAX_X
  FROM sample_polys
```

結果:

```
OVERALL_MAX_X
--------------
          120
```

例 4

この例は、SAMPLE_POLYS 表にあるすべてのポリゴンの空間のエクステント (全体としての最小から全体としての最大) を見つけます。この計算は通常、データを追加する余裕があるかどうかを判断するため、データに関連付けられた空間参照系の空間エクステントと、形状の実際の空間エクステントを比較するために使用されます。

```
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
```

結果:

```
MIN_X     MIN_Y     MIN_Z     MIN_M     MAX_X     MAX_Y     MAX_Z     MAX_M
--------- --------- --------- --------- --------- --------- --------- ----------
        0         0        10         3       120       140        40         16
```

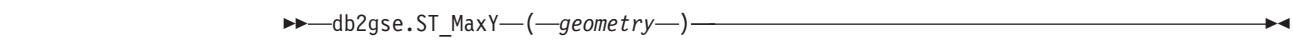
ST_MaxY

ST_MaxY は形状を入力パラメーターとし、その最大 Y 座標を戻します。

与えられた形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
最大 Y 座標を戻す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

DOUBLE

例

例 1

この例は、ST_MaxY 関数の使用法を示しています。3 つのポリゴンを作成し、SAMPLE_POLYS 表に挿入します。

```
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) )
```

例 2

この例は、SAMPLE_POLYS 内の各ポリゴンの最大の Y 座標を見つけます。

```
SELECT id, CAST ( ST_MaxY(geometry) AS INTEGER) MAX_Y
  FROM sample_polys
```

結果:

```
ID          MAX_Y
----------- ------------
          1          140
          2            4
          3           13
```

例 3

この例は、GEOMETRY 列のすべてのポリゴンに対して存在する最大の Y 座標を見つけます。

```
SELECT CAST ( MAX ( ST_MaxY(geometry) ) AS INTEGER) OVERALL_MAX_Y
  FROM sample_polys
```

結果:

```
OVERALL_MAX_Y
--------------
          140
```

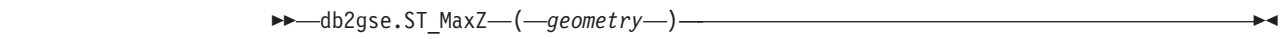
ST_MaxZ

ST_MaxZ は形状を入力パラメーターとし、その最大 Z 座標を戻します。

与えられた形状が NULL または空の場合、または Z 座標がない場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
最大 Z 座標を戻す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

DOUBLE

例

例 1

この例は、ST_MaxZ 関数の使用法を示しています。3 つのポリゴンを作成し、SAMPLE_POLYS 表に挿入します。

```
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) )
```

例 2

この例は、SAMPLE_POLYS 内の各ポリゴンの最大の Z 座標を見つけます。

```
SELECT id, CAST ( ST_MaxZ(geometry) AS INTEGER) MAX_Z
  FROM sample_polys
```

結果:

```
ID          MAX_Z
----------- ------------
          1           26
          2           40
          3           12
```

例 3

この例は、GEOMETRY 列のすべてのポリゴンに対して存在する最大の Z 座標を見つけます。

```
SELECT CAST ( MAX ( ST_MaxZ(geometry) ) AS INTEGER) OVERALL_MAX_Z
  FROM sample_polys
```

結果:

```
OVERALL_MAX_Z
--------------
            40
```

ST_MBR

ST_MBR は形状を入力パラメーターとし、その最小外接長方形を戻します。

与えられた形状がポイントの場合は、ポイント自体が戻されます。形状が水平線または垂直線で、空間参照系が測地でない場合、水平線または垂直線自体が戻されます。それ以外の場合、最小外接長方形がポリゴンとして戻されます。与えられた形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
最小外接長方形を戻す形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

db2gse.ST_Geometry

例

この例は、ST_MBR 関数を使用して、ポリゴンの最小外接長方形を戻す方法を示しています。指定された形状はポリゴンなので、最小外接長方形はポリゴンとして戻されます。

次の例で、結果の線は読みやすくするために再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

```
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
```

結果:

```
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 ))
```

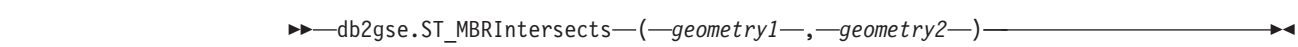
ST_MBRIntersects

ST_MBRIntersects は 2 つの形状を入力パラメーターとし、2 つの形状の最小外接長方形が交差する場合は 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。ポイントおよび、水平または垂直の折れ線の最小外接長方形は、形状そのものです。

2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。

与えられた形状のいずれかが NULL または空の場合は、NULL が戻されます。

構文



パラメーター

geometry1
その形状の最小外接長方形が *geometry2* の最小外接長方形と交差するかどうかをテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2
その形状の最小外接長方形が *geometry1* の最小外接長方形と交差するかどうかをテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

例 1

この例は、ST_MBRIntersects を使用して、2 つの交差していないポリゴンの最小外接長方形が交差するかどうかを調べることによって、互いに近くにあるかどうかを概算する方法を示しています。最初の例は SQL CASE 式を使用しています。2 番目の例は、1 つの SELECT ステートメントを使用して、ID = 2 を持つポリゴンの最小外接長方形と交差するポリゴンを見つけます。

```
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) )
```

例 2

次の SELECT ステートメントは、CASE 式を使用して、お互いに交差する最小外接長方形を持つポリゴンの ID を見つけます。

```
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
```

結果:

```
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
```

例 3

次の SELECT ステートメントは、形状の最小外接長方形が、ID = 2 を持つポリゴンの最小外接長方形と交差するかどうかを判別しています。

```
SELECT a.id, b.id, ST_MBRIntersects (a.geometry, b.geometry) MBR_INTERSECTS
  FROM sample_polys a, sample_polys b
  WHERE a.id = 2
```

結果:

```
ID         ID           MBR_INTERSECTS
---------- ------------ ---------------
         2            1               1
         2            2               1
         2            3               0
```

ST_MeasureBetween または ST_LocateBetween

ST_MeasureBetween または ST_LocateBetween は、形状および 2 つの M 座標 (指標) を入力パラメーターとし、その形状の、2 つの M 座標の間の切断されたパスまたはポイントのセットを表す部分を戻します。

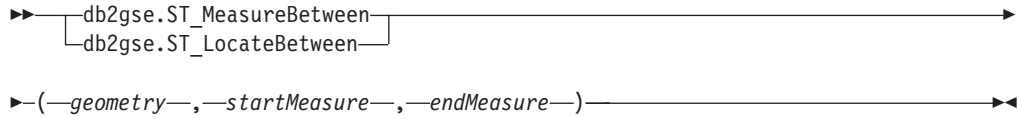
曲線、複数曲線、面、および複数面の場合、結果を計算するために補間が行われます。結果の形状は、与えられた形状の空間参照系で表現されます。

与えられた形状が面または複数面の場合、ST_MeasureBetween または ST_LocateBetween は形状の外部および内部リングに適用されます。与えられた形状のいずれの部分も与えられた M 座標で定義されたインターバルにない場合は、空の形状が戻されます。与えられた形状が NULL の場合は NULL が戻されます。

結果の形状が空でない場合、複数ポイントまたは複数折れ線のタイプが戻されます。

どちらの関数も、メソッドとして呼び出すことができます。

構文



パラメーター

geometry
指標値が *startMeasure* から *endMeasure* である部分がある形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

startMeasure
測定インターバルの下限を表すDOUBLE タイプの値。この値が NULL の場合、下限は適用されません。

endMeasure
測定インターバルの上限を表すDOUBLE タイプの値。この値が NULL の場合、上限は適用されません。

戻りタイプ

db2gse.ST_Geometry

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

形状の M 座標 (指標) はユーザーにより定義されます。指標は、測定したいものを何でも表現できる (例えば、高速道路の距離、温度、圧力、pH など) ので、千差万別です。

この例は、pH を測定して収集したデータを記録するための M 座標の使用を説明しています。研究者は特定の場所の高速道路に沿った土壌の pH を収集しています。通常の操作手順にしたがって、研究者は土壌サンプルを採取した場所ごとに必要な値 (採取した場所の X 座標、Y 座標、および測定した pH 値) を書き留めます。

```
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 ) )
```

土壌の酸性が 4 から 6 の間で変化するパスを見つけるには、研究者は次の SELECT ステートメントを使用します。

```
SELECT id, CAST( ST_AsText( ST_MeasureBetween( 4, 6) )
  AS VARCHAR(150) ) MEAS_BETWEEN_4_AND_6
  FROM sample_lines
```

結果:

```
ID         MEAS_BETWEEN_4_AND_6
---------- ----------------------------------------------------
         1 LINESTRING M (3.00000000 4.33333300 4.00000000,
                         3.00000000 3.00000000 6.00000000,
                         4.00000000 4.00000000 6.00000000,
                         5.00000000 5.00000000 6.00000000)
```

ST_MidPoint

ST_MidPoint は曲線を入力パラメーターとし、曲線に沿って測定して、曲線の両端から等距離にある曲線上のポイントを戻します。結果のポイントは、与えられた曲線の空間参照系で表現されます。

与えられた曲線が空の場合は、空のポイントが戻されます。与えられた曲線が NULL の場合は、NULL が戻されます。

曲線が Z 座標または M 座標 (指標) を含む場合、中点は曲線内の X および Y 座標の値のみによって決められます。戻されたポイントの Z 座標および指標は、補間されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

curve その中心を戻す曲線を表す、ST_Curve タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

db2gse.ST_Point

例

この例は、曲線の中点を戻させるための ST_MidPoint の使用を示しています。

```
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
```

結果:

```
ID         MID_POINT
---------- -----------------------------------
         1 POINT ( 0.00000000 20.00000000)
         2 POINT ( 3.00000000 3.45981800)
         3 POINT ( 5.00000000 0.00000000)
         4 POINT ( 7.50000000 20.00000000)
```

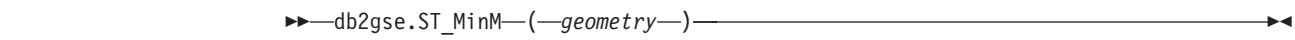
ST_MinM

ST_MinM は形状を入力パラメーターとし、その最小 M 座標を戻します。

与えられた形状が NULL または空の場合、または M 座標がない場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
最小 M 座標を戻す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

DOUBLE

例

例 1

この例は、ST_MinM 関数の使用法を示しています。3 つのポリゴンを作成し、SAMPLE_POLYS 表に挿入します。

```
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) )
```

例 2

この例は、SAMPLE_POLYS 内の各ポリゴンの最小の M 座標を見つけます。

```
SELECT id, CAST ( ST_MinM(geometry) AS INTEGER) MIN_M
  FROM sample_polys
```

結果:

```
ID          MIN_M
----------- -----------
          1           3
          2           5
          3          11
```

例 3

この例は、GEOMETRY 列のすべてのポリゴンに対して存在する、最小の M 座標を見つけます。

```
SELECT CAST ( MIN ( ST_MinM(geometry) ) AS INTEGER) OVERALL_MIN_M
  FROM sample_polys
```

結果:

```
OVERALL_MIN_M
-------------
            3
```

ST_MinX

ST_MinX は形状を入力パラメーターとし、その最小 X 座標を戻します。

与えられた形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
最小 X 座標を戻す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

DOUBLE

例

例 1

この例は、ST_MinX 関数の使用法を示しています。3 つのポリゴンを作成し、SAMPLE_POLYS 表に挿入します。

```
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) )
```

例 2

この例は、SAMPLE_POLYS 内の各ポリゴンの最小の X 座標を見つけます。

```
SELECT id, CAST ( ST_MinX(geometry) AS INTEGER) MIN_X
  FROM sample_polys
```

結果:

```
ID          MIN_X
----------- ------------
          1          110
          2            0
          3            8
```

例 3

この例は、GEOMETRY 列内のすべてのポリゴンに対して存在する、最小の X 座標を見つけます。

```
SELECT CAST ( MIN ( ST_MinX(geometry) ) AS INTEGER) OVERALL_MIN_X
  FROM sample_polys
```

結果:

```
OVERALL_MIN_X
--------------
             0
```

ST_MinY

ST_MinY は形状を入力パラメーターとし、その最小 Y 座標を戻します。

与えられた形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
最小 Y 座標を戻す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

DOUBLE

例

例 1

この例は、ST_MinY 関数の使用法を示しています。3 つのポリゴンを作成し、SAMPLE_POLYS 表に挿入します。

```
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) )
```

例 2

この例は、SAMPLE_POLYS 内の各ポリゴンの最小の Y 座標を見つけます。

```
SELECT id, CAST ( ST_MinY(geometry) AS INTEGER) MIN_Y
  FROM sample_polys
```

結果:

```
ID          MIN_Y
----------- ------------
          1          120
          2            0
          3            4
```

例 3

この例は、GEOMETRY 列内のすべてのポリゴンについて存在する、最小の Y 座標を見つけます。

```
SELECT CAST ( MIN ( ST_MinY(geometry) ) AS INTEGER) OVERALL_MIN_Y
  FROM sample_polys
```

結果:

```
OVERALL_MIN_Y
-------------
            0
```

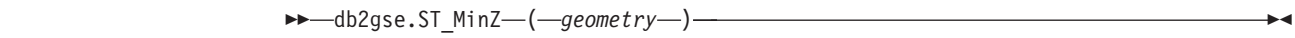
ST_MinZ

ST_MinZ は形状を入力パラメーターとし、その最小 Z 座標を戻します。

与えられた形状が NULL または空の場合、または Z 座標がない場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
最小 Z 座標を戻す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

DOUBLE

例

例 1

この例は、ST_MinZ 関数の使用法を示しています。3 つのポリゴンを作成し、SAMPLE_POLYS 表に挿入します。

```
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) )
```

例 2

この例は、SAMPLE_POLYS 内の各ポリゴンの最小の Z 座標を見つけます。

```
SELECT id, CAST ( ST_MinZ(geometry) AS INTEGER) MIN_Z
  FROM sample_polys
```

結果:

```
ID          MIN_Z
----------- ------------
          1           20
          2           31
          3           10
```

例 3

この例は、GEOMETRY 列内のすべてのポリゴンについて存在する、最小の Z 座標を見つけます。

```
SELECT CAST ( MIN ( ST_MinZ(geometry) ) AS INTEGER) OVERALL_MIN_Z
  FROM sample_polys
```

結果:

```
OVERALL_MIN_Z
--------------
            10
```

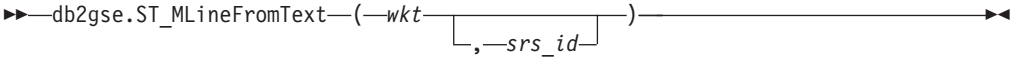
ST_MLineFromText

ST_MLineFromText は、複数折れ線の事前割り当てテキスト表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応する複数折れ線を戻します。

指定された事前割り当てテキスト表記が NULL の場合は、NULL が戻されます。

同じ結果を得るには ST_MultiLineString 関数を使用することをお勧めします。お勧めする理由は、ST_MultiLineString は事前割り当てテキスト表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkt 結果の複数折れ線の事前割り当てテキスト表記が入る、CLOB(2G) タイプの値。

srs_id 結果の複数折れ線の空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

指定された *srs_id* が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が発生します (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_MultiLineString

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_MLineFromText を使用して、事前割り当てテキスト表記から複数折れ線を作成し、挿入する方法を示しています。挿入されるレコードの ID は 1110 で、形状は空間参照系 1 の複数折れ線です。複数折れ線は、複数折れ線の事前割り当てテキスト表記になっています。この形状の X および Y 座標は以下のとおりです。

- Line 1: (33, 2) (34, 3) (35, 6)
- Line 2: (28, 4) (29, 5) (31, 8) (43, 12)
- Line 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) )
```

次の SELECT ステートメントは、表に記録された複数折れ線を戻します。

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(280) ) MULTI_LINE_STRING
  FROM sample_mlines
  WHERE id = 1110
```

結果:

```
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 ))
```

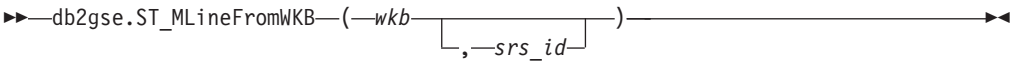
ST_MLineFromWKB

ST_MLineFromWKB は、複数折れ線の事前割り当てバイナリー表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応する複数折れ線を戻します。

与えられた、事前割り当てバイナリー表記が NULL の場合は、NULL が戻されます。

同じ結果を得るには ST_MultiLineString 関数を使用することをお勧めします。お勧めする理由は、ST_MultiLineString は事前割り当てバイナリー表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkb 結果の複数折れ線の事前割り当てバイナリー表記が入る、BLOB(2G) タイプの値。

srs_id 結果の複数折れ線の空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

指定された *srs_id* が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が発生します (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_MultiLineString

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_MLineFromWKB を使用して事前割り当てバイナリー表記から複数折れ線を作成する方法を示しています。この形状は空間参照系 1 の複数折れ線です。この例では、複数折れ線は ID = 10 を使用して、SAMPLE_MLINES 表の GEOMETRY 列に保管され、WKB 列が ST_AsBinary 関数を使用して事前割り当てバイナリー表記で更新されます。最後に ST_MLineFromWKB 関数を使用して、WKB 列から複数折れ線を戻します。この形状の X および Y 座標は以下のとおりです。

- Line 1: (61, 2) (64, 3) (65, 6)
- Line 2: (58, 4) (59, 5) (61, 8)
- Line 3: (69, 3) (67, 4) (66, 7) (68, 9)

SAMPLE_MLINES 表には、複数折れ線が保管されている GEOMETRY 列と、複数折れ線の事前割り当てバイナリー表記が保管されている WKB 列があります。

```
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
```

次の SELECT ステートメントでは、ST_MLineFromWKB 関数を使用して WKB 列から複数折れ線を検索しています。

```
SELECT id, CAST( ST_AsText( ST_MLineFromWKB (wkb) )
       AS VARCHAR(280) ) MULTI_LINE_STRING
  FROM sample_mlines
  WHERE id = 10
```

結果:

```
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 ))
```

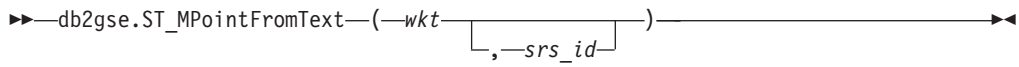
ST_MPointFromText

ST_MPointFromText は、複数ポイントの事前割り当てテキスト表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応する複数ポイントを戻します。

指定された事前割り当てテキスト表記が NULL の場合は、NULL が戻されます。

同じ結果を得るには、ST_MultiPoint 関数をお勧めします。その理由は、ST_MultiPoint は事前割り当てテキスト表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkt 結果の複数ポイントの事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。

srs_id 結果の複数ポイントの空間参照系を識別する、タイプ INTEGER の値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

指定された *srs_id* が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が発生します (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_MultiPoint

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は ST_MPointFromText を使用して、事前割り当てテキスト表記から複数ポイントを作成し、挿入する方法を示しています。挿入されるレコードの ID は 1110 で、形状は空間参照系 1 の複数ポイントです。複数ポイントは、複数ポイントの事前割り当てテキスト表記になっています。この形状の X 座標と Y 座標は、(1, 2) (4, 3) (5, 6) です。

```
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) )
```

次の SELECT ステートメントは、表に記録された複数ポイントを戻します。

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(280) ) MULTIPOINT
  FROM sample_mpoints
  WHERE id = 1110
```

結果:

```
ID         MULTIPOINT
---------- ------------------------------------------------------------------
      1110 MULTIPOINT (1.00000000 2.00000000, 4.00000000 3.00000000,
                       5.00000000 6.00000000)
```

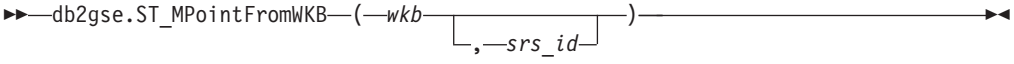
ST_MPointFromWKB

ST_MPointFromWKB は、複数ポイントの事前割り当てバイナリー表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応する複数ポイントを戻します。

与えられた、事前割り当てバイナリー表記が NULL の場合は、NULL が戻されます。

同じ結果を得るには、ST_MultiPoint 関数をお勧めします。お勧めする理由は、ST_MultiPoint は事前割り当てバイナリー表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkb 結果の複数ポイントの事前割り当てバイナリー表記が入る、タイプ BLOB(2G) の値。

srs_id 結果の複数ポイントの空間参照系を識別する、タイプ INTEGER の値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

指定された *srs_id* が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が発生します (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_MultiPoint

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_MPointFromWKB を使用して、事前割り当てバイナリー表記から複数ポイントを作成する方法を示しています。この形状は空間参照系 1 の複数ポイントです。この例では、複数ポイントは ID = 10 を使用して、SAMPLE_MPOINTS 表の GEOMETRY 列に保管され、WKB 列が ST_AsBinary 関数を使用して事前割り当てバイナリー表記で更新されます。最後に ST_MPointFromWKB 関数を使用して、WKB 列から複数ポイントを戻します。この形状の X 座標と Y 座標は、(44, 14) (35, 16) (24, 13) です。

SAMPLE_MPOINTS 表には、複数ポイントが保管されている GEOMETRY 列と、複数ポイントの事前割り当てバイナリー表記が保管されている WKB 列があります。

```
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
```

次の SELECT ステートメントでは、ST_MPointFromWKB 関数を使用して WKB 列から複数ポイントを検索しています。

```
SELECT id, CAST( ST_AsText( ST_MLineFromWKB (wkb)) AS VARCHAR(100)) MULTIPOINT
  FROM sample_mpoints
  WHERE id = 10
```

結果:

```
ID         MULTIPOINT
---------- ----------------------------------------------------------------
        10 MULTIPOINT (44.00000000 14.00000000, 35.00000000
              16.00000000 24.00000000 13.00000000)
```

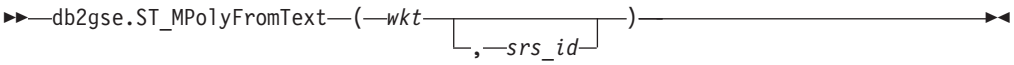
ST_MPolyFromText

ST_MPolyFromText は、複数ポリゴンの事前割り当てテキスト表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応する複数ポリゴンを戻します。

指定された事前割り当てテキスト表記が NULL の場合は、NULL が戻されます。

同じ結果を得るには、ST_MultiPolygon 関数をお勧めします。その理由は、ST_MultiPolygon は事前割り当てテキスト表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkt 結果の複数ポリゴンの事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。

srs_id 結果の複数ポリゴンの空間参照系を識別する、タイプ INTEGER の値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

指定された *srs_id* が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が発生します (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_MultiPolygon

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_MPolyFromText を使用して、事前割り当てテキスト表記から複数ポリゴンを作成し、挿入する方法を示しています。挿入されるレコードの ID は 1110 で、形状は空間参照系 1 の複数ポリゴンです。複数ポリゴンは、複数ポリゴンの事前割り当てテキスト表記になっています。この形状の X および Y 座標は以下のとおりです。

- Polygon 1: (3, 3) (4, 6) (5, 3) (3, 3)
- Polygon 2: (8, 24) (9, 25) (1, 28) (8, 24)
- Polygon 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) )
```

次の SELECT ステートメントは、表に記録された複数ポリゴンを戻します。

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(350) ) MULTI_POLYGON
  FROM sample_mpolys
  WHERE id = 1110
```

結果:

```
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)))
```

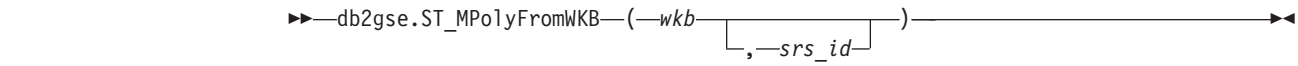
ST_MPolyFromWKB

ST_MPolyFromWKB は、複数ポリゴンの事前割り当てバイナリー表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応する複数ポリゴンを戻します。

与えられた、事前割り当てバイナリー表記が NULL の場合は、NULL が戻されます。

同じ結果を得るには、ST_MultiPolygon 関数をお勧めします。お勧めする理由は、ST_MultiPolygon は事前割り当てバイナリー表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkb 結果の複数ポリゴンの事前割り当てバイナリー表記が入る、タイプ BLOB(2G) の値。

srs_id 結果の複数ポリゴンの空間参照系を識別する、タイプ INTEGER の値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

指定された *srs_id* が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が発生します (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_MultiPolygon

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例では、ST_MPolyFromWKB を使用して、事前割り当てバイナリー表記から複数ポリゴンを作成する方法を示しています。この形状は空間参照系 1 の複数ポリゴ

ンです。この例では、複数ポリゴンは ID = 10 を使用して、SAMPLE_MPOLYS 表の GEOMETRY 列に保管され、次に ST_AsBinary 関数を使用して WKB 列を事前割り当てバイナリー表記で更新しています。最後に ST_MPolyFromWKB 関数を使用して、WKB 列から複数ポリゴンを戻します。この形状の X および Y 座標は以下のとおりです。

- Polygon 1: (1, 72) (4, 79) (5, 76) (1, 72)
- Polygon 2: (10, 20) (10, 40) (30, 41) (10, 20)
- Polygon 3: (9, 43) (7, 44) (6, 47) (9, 43)

SAMPLE_MPOLYS 表には、複数ポリゴンが保管される GEOMETRY 列と、複数ポリゴンの事前割り当てバイナリー表記が保管される WKB 列があります。

```
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
```

次の SELECT ステートメントでは、ST_MPolyFromWKB 関数を使用して WKB 列から複数ポリゴンを検索しています。

```
SELECT id, CAST( ST_AsText( ST_MPolyFromWKB (wkb) )
    AS VARCHAR(320) ) MULTIPOLYGON
  FROM sample_mpolys
  WHERE id = 10
```

結果:

```
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 )))
```

ST_MultiLineString

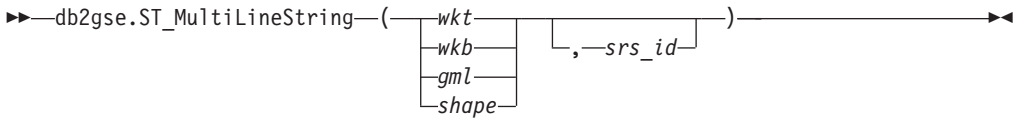
ST_MultiLineString は、次の入力の 1 つから複数折れ線を作成します。

- 事前割り当てテキスト表記
- 事前割り当てバイナリー表記
- 形状表記
- Geography Markup Language (GML) 表記

結果の複数折れ線を入れる空間参照系を示すため、オプションとして空間参照系 ID を指定することができます。

事前割り当てテキスト表記、事前割り当てバイナリー表記、形状表記、または GML 表記が NULL の場合は、NULL が戻されます。

構文



パラメーター

wkt 結果の複数折れ線の事前割り当てテキスト表記が入る、CLOB(2G) タイプの値。

wkb 結果の複数折れ線の事前割り当てバイナリー表記が入る、BLOB(2G) タイプの値。

gml Geography Markup Language (GML) を使用した、結果の複数折れ線を表す、CLOB(2G) タイプの値。

shape 結果の複数折れ線の形状表記を表す、BLOB(2G) タイプの値。

srs_id 結果の複数折れ線の空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_MultiLineString

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_MultiLineString を使用して、事前割り当てテキスト表記から複数折れ線を作成し、挿入します。挿入されるレコードの ID は 1110 で、形状は空間参照系 1 の複数折れ線です。複数折れ線は、複数折れ線の事前割り当てテキスト表記になっています。この形状の X および Y 座標は以下のとおりです。

- Line 1: (33, 2) (34, 3) (35, 6)
- Line 2: (28, 4) (29, 5) (31, 8) (43, 12)
- Line 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) )
```

次の SELECT ステートメントは、表に記録された複数折れ線を戻します。

```
SELECT id,
       CAST( ST_AsText( geometry ) AS VARCHAR(280) )
  MULTI_LINE_STRING
  FROM sample_mlines
  WHERE id = 1110
```

結果:

```
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 ))
```

ST_MultiPoint

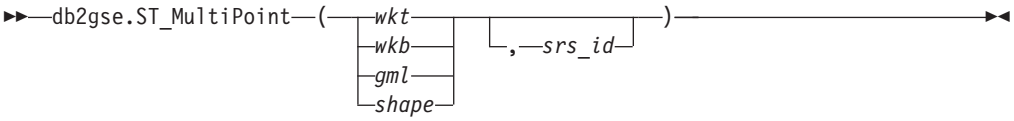
ST_MultiPoint は、以下の入力の 1 つから複数ポイントを作成します。

- 事前割り当てテキスト表記
- 事前割り当てバイナリー表記
- 形状表記
- Geography Markup Language (GML) 表記

結果の複数ポイントを入れる空間参照系を示すため、オプションとして空間参照系 ID を指定することができます。

事前割り当てテキスト表記、事前割り当てバイナリー表記、形状表記、または GML 表記が NULL の場合は、NULL が戻されます。

構文



パラメーター

wkt 結果の複数ポイントの事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。

wkb 結果の複数ポイントの事前割り当てバイナリー表記が入る、タイプ BLOB(2G) の値。

gml Geography Markup Language (GML) を使用した、結果の複数ポイントを表す、CLOB(2G) タイプの値。

shape 結果の複数ポイントの形状表記を表す、BLOB(2G) タイプの値。

srs_id 結果の複数ポイントの空間参照系を識別する、タイプ INTEGER の値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_Point

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は ST_MultiPoint を使用して、事前割り当てテキスト表記から複数ポイントを作成し、挿入します。挿入されるレコードの ID は 1110 で、形状は空間参照系 1 の複数ポイントです。複数ポイントは、複数ポイントの事前割り当てテキスト表記になっています。この形状の X 座標と Y 座標は、(1, 2) (4, 3) (5, 6) です。

```
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))
```

次の SELECT ステートメントは、表に記録された複数ポイントを戻します。

```
SELECT id, CAST( ST_AsText(geometry) AS VARCHAR(90)) MULTIPOINT
  FROM sample_mpoints
  WHERE id = 1110
```

結果:

```
ID          MULTIPOINT
----------- ----------------------------------------------------
      1110 MULTIPOINT (1.00000000 2.00000000, 4.00000000
           3.00000000, 5.00000000 6.00000000)
```

ST_MultiPolygon

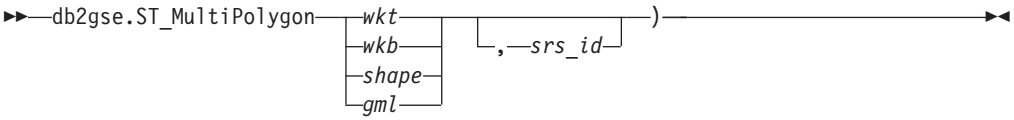
ST_MultiPolygon は、次の入力の 1 つから複数ポリゴンを作成します。

- 事前割り当てテキスト表記
- 事前割り当てバイナリー表記
- 形状表記
- Geography Markup Language (GML) 表記

結果の複数ポリゴンを入れる空間参照系を示すため、オプションとして空間参照系 ID を指定することができます。

事前割り当てテキスト表記、事前割り当てバイナリー表記、形状表記、または GML 表記が NULL の場合は、NULL が戻されます。

構文



パラメーター

wkt 結果の複数ポリゴンの事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。

wkb 結果の複数ポリゴンの事前割り当てバイナリー表記が入る、タイプ BLOB(2G) の値。

gml Geography Markup Language (GML) を使用した、結果の複数ポリゴンを表す、CLOB(2G) タイプの値。

shape 結果の複数ポリゴンの形状表記を表す、BLOB(2G) タイプの値。

srs_id 結果の複数ポリゴンの空間参照系を識別する、タイプ INTEGER の値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_MultiPolygon

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_MultiPolygon を使用して、事前割り当てテキスト表記から複数ポリゴンを作成し、挿入しています。挿入されるレコードの ID は 1110 で、形状は空間参照系 1 の複数ポリゴンです。複数ポリゴンは、複数ポリゴンの事前割り当てテキスト表記になっています。この形状の X および Y 座標は以下のとおりです。

- Polygon 1: (3, 3) (4, 6) (5, 3) (3, 3)
- Polygon 2: (8, 24) (9, 25) (1, 28) (8, 24)
- Polygon 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) )
```

次の SELECT ステートメントは、表に記録された複数ポリゴンを戻します。

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(350) ) MULTI_POLYGON
  FROM sample_mpolys
  WHERE id = 1110
```

結果:

```
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)))
```

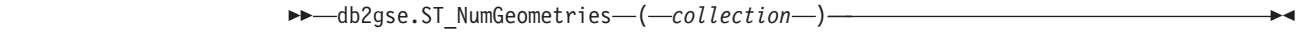
ST_NumGeometries

ST_NumGeometries は形状集合を入力パラメーターとし、その集合内にある形状の数を戻します。

与えられた形状集合が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

collection
形状の数を戻す形状集合を表す、ST_GeomCollection タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

2 つの形状集合を SAMPLE_GEOMCOLL 表に保管します。1 つは複数ポリゴンで、もう 1 つは複数ポイントです。ST_NumGeometries 関数は、それぞれの形状集合内に個々の形状がいくつあるかを判別します。

```
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
```

結果:

```
ID          NUM_GEOMS_IN_COLL
----------- -----------------
          1                 3
          2                 5
```

ST_NumInteriorRing

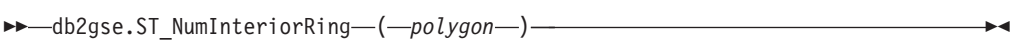
ST_NumInteriorRing はポリゴンを入力パラメーターとし、その内部リングの数を戻します。

与えられたポリゴンが NULL または空の場合は、NULL が戻されます。

ポリゴンが内部リングを持たない場合は、0 (ゼロ) が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

polygon
内部リングの数を戻すポリゴンを表す値 (タイプは ST_Polygon)。

戻りタイプ

INTEGER

例

以下の例は、次の 2 つのポリゴンを作成します。

- 内部リングを 2 つ持つもの
- 内部リングを持たないもの

```
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) )
```

ST_NumInteriorRing 関数を使用して、表の中の形状内のリングの数を戻します。

```
SELECT id, ST_NumInteriorRing(geometry) NUM_RINGS
FROM sample_polys
```

結果:

```
ID         NUM_RINGS
---------- --------------
         1              2
         2              0
```

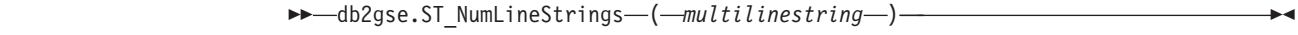
ST_NumLineStrings

ST_NumLineStrings は複数折れ線を入力パラメーターとし、その中に含まれている折れ線の数を戻します。

与えられた複数折れ線が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

multilinestring
折れ線の数を戻す複数折れ線を表す、ST_MultiLineString タイプの値。

戻りタイプ

INTEGER

例

複数折れ線を SAMPLE_MLINES 表に保管します。ST_NumLineStrings 関数を使用して、それぞれの複数折れ線内に個々の形状がいくつあるかを判別します。

```
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
```

結果:

```
ID          NUM_WITHIN
----------- ----------
        110          3
        111          2
```

ST_NumPoints

ST_NumPoints は、形状を入力パラメーターとし、その形状を定義するために使用されたポイントの数を戻します。例えば、形状がポリゴンであり、そのポリゴンを定義するために 5 つのポイントが使用されている場合、戻される数は 5 です。

与えられた形状が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry
ポイントの数を戻す形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

表に各種の形状を保管します。ST_NumPoints 関数により、SAMPLE_GEOMETRIES 表の各形状の中にあるポイントの数を判別します。

```
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
```

結果:

```
SPATIAL_TYPE    NUM_POINTS
--------------- ----------
st_point                 1
st_linestring            3
st_polygon               4
```

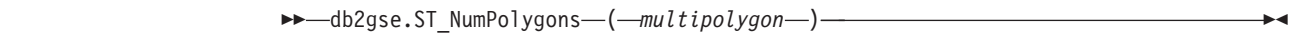
ST_NumPolygons

ST_NumPolygons は複数ポリゴンを入力パラメーターとし、その中に含まれているポリゴンの数を戻します。

与えられた複数ポリゴンが NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

multipolygon
ポリゴンの数を戻す複数ポリゴンを表す、ST_MultiPolygon タイプの値。

戻りタイプ

INTEGER

例

複数ポリゴンを SAMPLE_MPOLYS 表に保管します。ST_NumPolygons 関数により、それぞれの複数ポリゴン内に個々の形状がいくつあるかを判別します。

```
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
```

結果:

```
ID          NUM_WITHIN
----------- ----------
          1          3
          2          0
          3          2
```

ST_Overlaps

ST_Overlaps は 2 つの形状を入力パラメーターとし、その 2 つの形状の交差が同じディメンションの形状になるが、そのどちらとも等しくない形状になった場合に 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

2 つの形状のいずれかが NULL または空の場合は、NULL が戻されます。

2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。

構文

パラメーター

geometry1
geometry2 とのオーバーラップをテストされる形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2
geometry1 とのオーバーラップをテストされる形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

例 1

この例は、ST_Overlaps の使用法を示しています。各種の形状を作成し、SAMPLE_GEOMETRIES 表に挿入します。

```
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) )
```

例 2

この例は、オーバーラップするポイントの ID を見つけます。

```
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
```

結果:

```
ID          ID         OVERLAP
----------- ---------- ------------------------
          1          1 Points_do_not_overlap
          2          1 Points_do_not_overlap
          2          2 Points_do_not_overlap
```

例 3

この例は、オーバーラップする線の ID を見つけます。

```
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
```

結果:

```
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
```

例 4

この例は、オーバーラップするポリゴンの ID を見つけます。

```
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
```

結果:

```
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
```

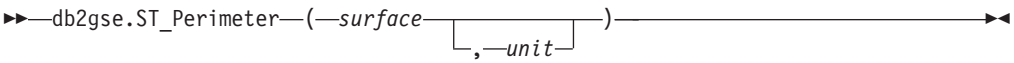
ST_Perimeter

ST_Perimeter は、面または複数面、およびオプションで単位を入力パラメーターとして取り、デフォルト、あるいは与えられた単位で、面または複数面の周囲の長さ (境界の長さ) を戻します。

与えられた面または複数面が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

surface
周囲の長さを戻す、ST_Surface タイプ、ST_MultiSurface タイプ、またはそのサブタイプの 1 つの値。

unit 周囲の長さを測定する単位を示す、VARCHAR(128) 値。サポートされる測定単位は DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビューにリストされています。

unit パラメーターを省略すると、次の規則を使用して perimeter を測定する単位が決められます。

- *surface* が投影座標システム、または地心から見た座標システムを使用する場合、この座標システムに関連付けられた線形単位がデフォルトになります。
- *surface* が地理座標システムで、測地座標システム (SRS) でない場合、この座標システムに関連付けられた角度単位がデフォルトになります。
- *surface* が測地 SRS の場合、デフォルトの測定単位はメートルになります。

単位変換の制約事項 : 以下の条件に当てはまる場合には、エラー (SQLSTATE 38SU4) が戻されます。

- 形状の座標システムが指定されておらず、かつ *unit* パラメーターが指定されている。
- 形状の座標システムが投影座標システムで、かつ角度単位が指定されている。
- 形状の座標システムが地理座標システムで、測地 SRS でなく、かつ線形単位が指定されている。
- 形状の座標システムが地理座標システムかつ測地 SRS であり、さらに角度単位が指定されている。

戻りタイプ

DOUBLE

例

例 1

この例は、ST_Perimeter 関数の使用法を示しています。db2se に対する呼び出しを使用して ID が 4000 の空間参照系を作成し、その空間参照系にポリゴンを作成します。

```
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
```

SAMPLE_POLYS 表を作成し、周囲の長さ 18 の形状を入れます。

```
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))
```

例 2

この例は、ポリゴンの ID および周囲の長さをリストします。

```
SELECT id, ST_Perimeter (geometry) AS PERIMETER
  FROM sample_polys
```

結果:

```
ID          PERIMETER
----------- ------------------------
          1 +1.80000000000000E+001
```

例 3

この例は、ポリゴンの ID と、メートルで測定したポリゴンの周囲の長さをリストします。

```
SELECT id, ST_Perimeter (geometry, 'METER') AS PERIMETER_METER
  FROM sample_polys
```

結果:

```
ID          PERIMETER_METER
----------- ------------------------
          1 +5.48641097282195E+000
```

ST_PerpPoints

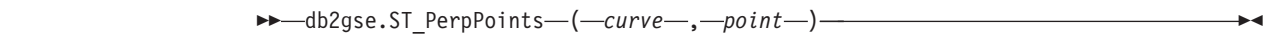
ST_PerpPoints は、曲線または複数曲線とポイントを入力パラメーターとし、その曲線または複数曲線上の与えられたポイントの垂直投影を戻します。与えられたポイントと垂直ポイントの間の距離が最小のポイントが戻されます。2 つ以上のこのような垂直に投影されたポイントが、与えられたポイントから等距離にある場合、それらすべてのポイントが戻されます。垂直のポイントが作成できない場合は、空のポイントが戻されます。

与えられた曲線または複数曲線が Z または M 座標を持つ場合、結果のポイントの Z または M 座標は、与えられた曲線または複数曲線を補間して計算されます。

与えられた曲線またはポイントが空の場合は、空のポイントが戻されます。与えられた曲線またはポイントが NULL の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

curve *point* の垂直投影を戻す曲線または複数曲線を表す、ST_Curve タイプまたは ST_MultiCurve タイプ、あるいはこれらのサブタイプの 1 つの値。

point *curve* 上に投影する垂直のポイントを表す、ST_Point タイプの値。

戻りタイプ

db2gse.ST_MultiPoint

例

例 1

この例は、ST_PerpPoints 関数を使用して、次の表に保管された折れ線に対して垂直であるポイントを検索する方法を示しています。INSERT ステートメントで ST_LineString 関数を使用し、折れ線を作成します。

```
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) )
```

例 2

この例は、座標 (5, 0) を持つポイントの折れ線上の垂直投影を見つけます。ST_AsText 関数を使用して、戻された値 (複数ポイント) をその事前割り当てテキスト表記に変換します。

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 0) ) )
  AS VARCHAR(50) ) PERP
  FROM sample_lines
```

結果:

```
PERP
----------------------------------------
MULTIPOINT ( 5.00000000 0.00000000)
```

例 3

この例は、座標 (5, 5) を持つポイントの折れ線上の垂直投影を見つけます。ここでは、与えられたロケーションから等距離にあるポイントが、折れ線上に 3 つあります。したがって、このすべてのポイントからなる複数ポイントが戻されます。

```
SELECT  CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 5) ) )
  AS VARCHAR160) ) PERP
  FROM sample_lines
```

結果:

```
PERP
--------------------------------------------------------------------------------
MULTIPOINT ( 0.00000000 5.00000000, 5.00000000 0.00000000, 10.00000000 5.00000000)
```

例 4

この例は、座標 (5, 10) を持つポイントの折れ線上の垂直投影を見つけます。ここでは、3 つの異なる垂直のポイントが見つかります。ただし、ST_PerpPoints 関数は、与えられたポイントに最も近いポイントだけを戻します。したがって、最も近い 2 つのポイントだけを含む複数ポイントが戻されます。3 番目のポイントは含まれません。

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 10) ) )
  AS VARCHAR(80) ) PERP
  FROM sample_lines
```

結果:

```
PERP
------------------------------------------------------------
MULTIPOINT ( 0.00000000 10.00000000, 10.00000000 10.00000000 )
```

例 5

この例は、座標 (5, 15) を持つポイントの折れ線上の垂直投影を見つけます。

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point('point(5 15)', 0 ) ) )
  AS VARCHAR(80) ) PERP
  FROM sample_lines
```

結果:

```
PERP
-----------------------------------
MULTIPOINT ( 5.00000000 0.00000000 )
```

例 6

この例では、座標 (15 15) を持つ指定されたポイントには、折れ線に垂直投影がありません。したがって、空の形状が戻されます。

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(15, 15) ) )
  AS VARCHAR(80) ) PERP
  FROM sample_lines
```

結果:

```
PERP
--------------------------------------------
MULTIPOINT EMPTY
```

ST_Point

ST_Point は、次の入力セットの 1 つからポイントを作成します。

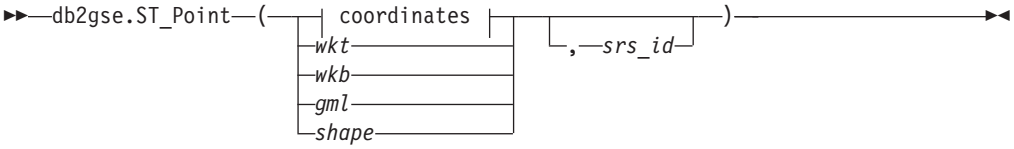
- X および Y 座標のみ
- X、Y、および Z 座標
- X、Y、Z、および M 座標

- 事前割り当てテキスト表記
- 事前割り当てバイナリー表記
- 形状表記
- Geography Markup Language (GML) 表記

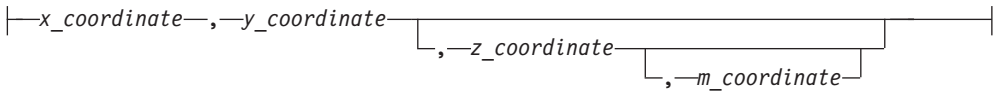
結果のポイントを置く空間参照系を示すため、オプションとして空間参照系 ID を指定することができます。

ポイントを座標から作成し、X または Y 座標が NULL の場合、例外条件が起こります (SQLSTATE 38SUP)。Z または M 座標が NULL の場合、結果のポイントは Z または M 座標を持ちません。事前割り当てテキスト表記、事前割り当てバイナリー表記、形状表記、または GML 表記からポイントを作成し、その表記が NULL の場合は、NULL が戻されます。

構文



座標:



パラメーター

wkt 結果のポイントの事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。

wkb 結果のポイントの事前割り当てバイナリー表記が入る、BLOB(2G) タイプの値。

gml Geography Markup Language (GML) を使用した、結果のポイントを表す、CLOB(2G) タイプの値。

shape 結果のポイントの形状表記を表す、BLOB(2G) タイプの値。

srs_id 結果のポイントの空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

x_coordinate
結果のポイントの X 座標を示す、DOUBLE タイプの値。

y_coordinate
結果のポイントの Y 座標を示す、DOUBLE タイプの値。

z_coordinate

結果のポイントの Z 座標を示す、DOUBLE タイプの値。

z_coordinate パラメーターを省略すると、結果のポイントは Z 座標を持ちません。このようなポイントの場合、ST_Is3D の結果は 0 (ゼロ) です。

m_coordinate

結果のポイントの M 座標を示す、DOUBLE タイプの値。

m_coordinate パラメーターを省略すると、結果のポイントは指標を持ちません。このようなポイントの場合 ST_IsMeasured の結果は 0 (ゼロ) です。

戻りタイプ

db2gse.ST_Point

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

例 1

この例は、ST_Point を使用してポイントを作成し、挿入する方法を示しています。最初のポイントは X と Y 座標のセットを使用して作成します。2 番目のポイントは、その事前割り当てテキスト表記を使用して作成します。ポイントは両方とも、空間参照系 1 の形状です。

```
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) )
```

次の SELECT ステートメントは、表に記録されたポイントを戻します。

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(90)) POINTS
  FROM sample_points
```

結果:

```
ID         POINTS
---------- -----------------------------------
      1110 POINT ( 10.00000000 20.00000000)
      1101 POINT ( 30.00000000 40.00000000)
```

例 2

この例は、X 座標 120、Y 座標 358、M 座標 34 を持ち、Z 座標を持たないポイントの値を、ID 1103 と一緒に、レコードとして SAMPLE_POINTS 表に挿入します。

```
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
```

結果:

```
ID         POINTS
 ---------- ------------------------------------------------
       1103 POINT M ( 120.0000000 358.0000000 34.00000000)
```

例 3

この例は、X 座標 1003、Y 座標 9876、Z 座標 20 を持ち、空間参照系 0 にあるポイントの値を、表記として Geography Markup Language (GML) を使用して、ID 1104 を持つ行として、SAMPLE_POINTS 表に挿入します。

```
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
```

結果:

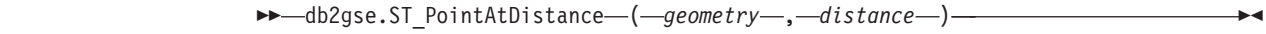
```
ID         POINTS
 ---------- ------------------------------------------------
       1104 POINT Z ( 1003.000000 9876.000000 20.00000000)
```

ST_PointAtDistance

ST_PointAtDistance は、入力パラメーターとして単一曲線または複数曲線の形状と 1 つの距離を取り、曲線形状に沿った所定の距離にあるポイント形状を返します。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
処理する形状を表す、ST_Curve タイプまたは ST_MultiCurve タイプまたはそのサブタイプの 1 つの値。

distance
ポイントを見つけるための、形状に沿った距離であるタイプ DOUBLE の値。

戻りタイプ

db2gse.ST_Point

例

例 1

以下の SQL ステートメントにより、2 つの列を持つ SAMPLE_GEOMETRIES 表が作成されます。ID 列で各行が一意的に識別されます。 GEOMETRY ST_LineString 列にはサンプル形状が格納されます。

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

以下の SQL ステートメントにより、2 つの行が SAMPLE_GEOMETRIES 表に挿入されます。

```
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))
```

以下の SELECT ステートメントおよび対応する結果セットは、ST_PointAtDistance 関数を使用して行ストリングの開始から 15 座標単位の距離にあるポイントを検出する方法を示しています。

```
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.
```

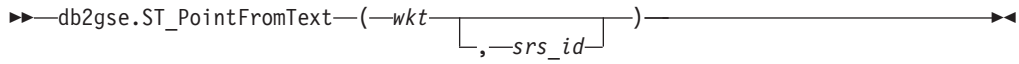
ST_PointFromText

ST_PointFromText は、ポイントの事前割り当てテキスト表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応するポイントを戻します。

指定された事前割り当てテキスト表記が NULL の場合は、NULL が戻されます。

同じ結果を得るには、ST_Point 関数をお勧めします。その理由は、ST_Point は事前割り当てテキスト表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkt 結果のポイントの事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。

srs_id 結果のポイントの空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_Point

例

この例は、ST_PointFromText を使用して、事前割り当てテキスト表記からポイントを作成し、挿入する方法を示しています。挿入されるレコードの ID は 1110 で、形状は空間参照系 1 のポイントです。ポイントは、ポイントの事前割り当てテキスト表記になっています。この形状の X および Y 座標は (10, 20) です。

```
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) )
```

次の SELECT ステートメントは、表に記録されたポリゴンを戻します。

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(35) ) POINTS
  FROM sample_points
  WHERE id = 1110
```

結果:

```
ID         POINTS
---------- -----------------------------------
      1110 POINTS ( 30.00000000 40.00000000)
```

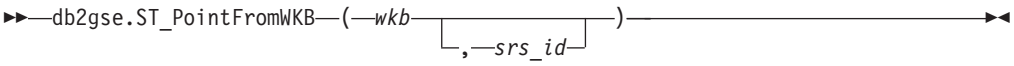
ST_PointFromWKB

ST_PointFromWKB は、ポイントの事前割り当てバイナリー表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応するポイントを戻します。

与えられた、事前割り当てバイナリー表記が NULL の場合は、NULL が戻されます。

同じ結果を得るには、ST_Point 関数をお勧めします。お勧めする理由は、ST_Point は事前割り当てバイナリー表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkb 結果のポイントの事前割り当てバイナリー表記が入る、BLOB(2G) タイプの値。

srs_id 結果のポイントの空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が暗黙的に使用されます。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_Point

例

この例は、ST_PointFromWKB を使用して、事前割り当てバイナリー表記からポイントを作成する方法を説明しています。形状は空間参照系 1 のポイントです。この例で、ポイントは SAMPLE_POLYS 表の GEOMETRY 列に保管され、次に WKB 列を事前割り当てバイナリー表記を使用して (ST_AsBinary 関数を使用) 更新しています。最後に ST_PointFromWKB 関数を使用して、WKB 列からポイントを戻します。

SAMPLE_POINTS 表には、ポイントが保管されている GEOMETRY 列と、そのポイントの事前割り当てバイナリー表記が保管される WKB 列があります。

```
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
```

次の SELECT ステートメントは、ST_PointFromWKB 関数を使用して、WKB 列からポイントを検索します。

```
SELECT id, CAST( ST_AsText( ST_PolyFromWKB (wkb) ) AS VARCHAR(35) ) POINTS
  FROM sample_points
```

結果:

```
ID         POINTS
---------- -----------------------------------
        10 POINT ( 44.00000000 14.00000000)
        11 POINT ( 24.00000000 13.00000000)
```

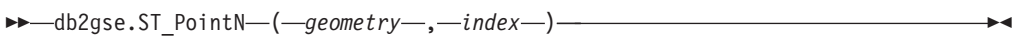
ST_PointN

ST_PointN は、折れ線または複数ポイントと索引を入力パラメーターとし、折れ線または複数ポイント内の、索引で指定されたポイントを戻します。結果のポイントは、与えられた折れ線または複数ポイントの空間参照系で表現されます。

与えられた折れ線または複数ポイントが NULL または空の場合は、NULL が戻されます。索引が 1 より小さいかまたは、折れ線または複数ポイント内のポイントの数より大きい場合は、NULL が戻され、警告条件 (SQLSTATE 01HS2) が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
index で指定されたポイントを戻す形状を表す、値 (タイプ ST_LineString または ST_MultiPoint)。

index *geometry* から戻される *n* 番目のポイントを指定する、INTEGER タイプの値。

戻りタイプ

db2gse.ST_Point

例

次の例は、ST_PointN の使用法を示しています。

```
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
```

結果:

```
ID        SECOND_INDEX
--------- ------------------------------
        1 POINT (5.00000000 5.00000000)
```

ST_PointOnSurface

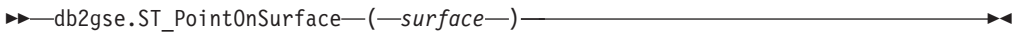
ST_PointOnSurface は面または複数面を入力パラメーターとし、面または複数面の内部にあることが保証されているポイントを戻します。このポイントは、面の準図心 (paracentroid) です。

結果のポイントは、与えられた面または複数面の空間参照系で表現されます。

与えられた面または複数面が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

surface

面上のポイントを戻す形状を表す、ST_Surface タイプ、ST_MultiSurface タイプ、またはそのサブタイプの 1 つの値。

戻りタイプ

db2gse.ST_Point

例

次の例は、2 つのポリゴンを作成してから、ST_PointOnSurface を使用しています。ポリゴンの 1 つはその中央に穴があります。戻されるポイントは、ポリゴンの面上にあります。これらは必ずしも正確にポリゴンの中央にあるわけではありません。

```
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
```

結果:

```
ID          POINT_ON_SURFACE
----------- ------------------------------------
          1 POINT ( 65.00000000 125.00000000)
          2 POINT ( 30.00000000 15.00000000)
```

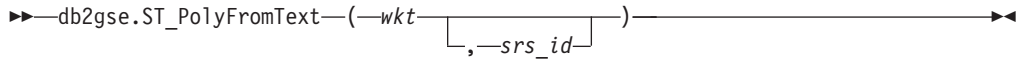
ST_PolyFromText

ST_PolyFromText は、ポリゴンの事前割り当てテキスト表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応するポリゴンを戻します。

指定された事前割り当てテキスト表記が NULL の場合は、NULL が戻されます。

同じ結果を得るには、ST_Polygon 関数をお勧めします。その理由は、ST_Polygon は事前割り当てテキスト表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkt 結果のポリゴンの事前割り当てテキスト表記が入る、CLOB(2G) タイプの値。

srs_id 結果のポリゴンの空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_Polygon

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_PolyFromText を使用して、事前割り当てテキスト表記からポリゴンを作成し、挿入する方法を示しています。挿入されるレコードの ID は 1110 で、形状は空間参照系 1 のポリゴンです。ポリゴンは、ポリゴンの事前割り当てテキスト表記になっています。この形状の X および Y 座標は (50, 20) (50, 40) (70, 30) です。

```
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) )
```

次の SELECT ステートメントは、表に記録されたポリゴンを戻します。

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(120) ) POLYGON
  FROM sample_polys
  WHERE id = 1110
```

結果:

```
ID          POLYGON
----------- ------------------------------------------------------------------
      1110 POLYGON (( 50.00000000 20.00000000, 70.00000000 30.00000000,
                      50.00000000 40.00000000, 50.00000000 20.00000000))
```

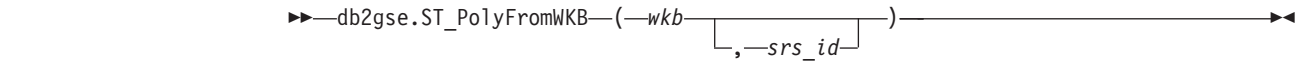
ST_PolyFromWKB

ST_PolyFromWKB は、ポリゴンの事前割り当てバイナリー表記および、オプションで空間参照系 ID を入力パラメーターとして取り、対応するポリゴンを戻します。

与えられた、事前割り当てバイナリー表記が NULL の場合は、NULL が戻されます。

同じ結果を得るには、ST_Polygon 関数をお勧めします。お勧めする理由は、ST_Polygon は事前割り当てバイナリー表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkb 結果のポリゴンの事前割り当てバイナリー表記が入る、BLOB(2G) タイプの値。

srs_id 結果のポリゴンの空間参照系を識別する、INTEGER タイプの値。

srs_id パラメーターを省略すると、数値 ID が 0 (ゼロ) の空間参照系が使用されます。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_Polygon

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_PolyFromWKB を使用して、事前割り当てバイナリー表記からポリゴンを作成する方法を示しています。この形状は空間参照系 1 のポリゴンです。この例では、ID = 1115 を使用して、SAMPLE_POLYS 表の GEOMETRY 列にポリゴンを保管し、(ST_AsBinary 関数を使用して) WKB 列を事前割り当てバイナリー表記で更新します。最後に ST_PolyFromWKB 関数を使用して、WKB 列から複数ポリゴンを戻します。この形状の X および Y 座標は (50, 20) (50, 40) (70, 30) です。

SAMPLE_POLYS 表は、ポリゴンが保管されている GEOMETRY 列と、そのポリゴンの事前割り当てバイナリー表記が保管されている WKB 列があります。

```
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
```

次の SELECT ステートメントは、ST_PolyFromWKB 関数を使用して WKB 列からポリゴンを検索しています。

```
SELECT id, CAST( ST_AsText( ST_PolyFromWKB (wkb) )
  AS VARCHAR(120) ) POLYGON
  FROM sample_polys
  WHERE id = 1115
```

結果:

```
ID         POLYGON
---------- ------------------------------------------------------------
      1115 POLYGON (( 50.00000000 20.00000000, 70.00000000
             30.00000000,50.00000000 40.00000000, 50.00000000
             20.00000000))
```

ST_Polygon

ST_Polygon は、次の入力の 1 つからポリゴンを作成します。

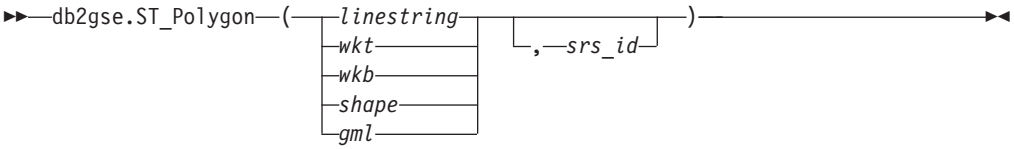
- 結果のポリゴンの外部リングを定義する、閉じた折れ線
- 事前割り当てテキスト表記
- 事前割り当てバイナリー表記
- 形状表記
- Geography Markup Language (GML) 表記

結果のポリゴンを入れる空間参照系を示すために、オプションとして空間参照系 ID を指定することができます。

ポリゴンが折れ線で構成され、与えられた折れ線が NULL の場合は、NULL が戻されます。与えられた折れ線が空の場合は、空のポリゴンが戻されます。ポリゴンが、ポリゴンの事前割り当てテキスト表記、事前割り当てバイナリー表記、形状表記、または GML 表記から構成され、その表記が NULL の場合は、NULL が戻されます。

この関数は、次の場合のみ、メソッドとして呼び出すこともできます。
ST_Polygon(*linestring*) および ST_Polygon(*linestring*,*srs_id*)。

構文



パラメーター

linestring
外側の境界用の外部リングを定義する折れ線を表す、ST_LineString タイプの値。*linestring* が閉じておらず、かつ単純な場合、例外条件が起こります (SQLSTATE 38SSL)。

wkt 結果のポリゴンの事前割り当てテキスト表記が入る、CLOB(2G) タイプの値。

wkb 結果のポリゴンの事前割り当てバイナリー表記が入る、BLOB(2G) タイプの値。

shape 結果のポリゴンの形状表記を表す、BLOB(2G) タイプの値。

gml Geography Markup Language (GML) を使用した、結果のポリゴンを表す、CLOB(2G) タイプの値。

srs_id 結果のポリゴンの空間参照系を識別する、INTEGER タイプの値。

ポリゴンが、与えられた *linestring* パラメーターで構成され、*srs_id* パラメーターが省略されている場合は、*linestring* からの空間参照系が暗黙的に使用されます。それ以外の場合、*srs_id* パラメーターが省略されていると、数値 ID 0 (ゼロ) の空間参照系が使用されます。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_Polygon

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_Polygon を使用してポリゴンを作成し、挿入する方法を示しています。3 つのポリゴンを作成し、挿入します。ポリゴンはすべて、空間参照系 1 の形状です。

- 最初のポリゴンはリング (閉じている、単純な折れ線) から作成されます。このポリゴンの X および Y 座標は (10, 20) (10, 40) (20, 30) です。
- 2 番目のポリゴンは、その事前割り当てテキスト表記を使用して作成されます。このポリゴンの X および Y 座標は (110, 120) (110, 140) (120, 130) です。
- 3 番目のポリゴンはドーナツ・ポリゴンです。ドーナツ・ポリゴンは 1 つの内部ポリゴンと 1 つの外部ポリゴンからなります。このドーナツ・ポリゴンは、その事前割り当てテキスト表記を使用して作成されます。外部ポリゴンの X および Y 座標は、(110, 120) (110, 140) (130, 140) (130, 120) (110, 120) です。内部ポリゴンの X および Y 座標は、(115, 125) (115, 135) (125, 135) (125, 135) (115, 125) です。

```
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))
```

次の SELECT ステートメントは、表に記録されたポリゴンを戻します。

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(120) ) POLYGONS
  FROM sample_polys
```

結果:

```
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))
```

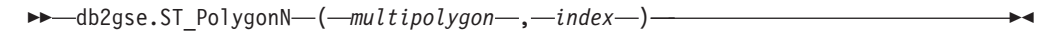
ST_PolygonN

ST_PolygonN は、複数ポリゴンと索引を入力パラメーターとし、索引で指定されたポリゴンを戻します。結果のポリゴンは、与えられた複数ポリゴンの空間参照系で表現されます。

与えられた複数ポリゴンが NULL または空の場合、または索引が 1 より小さいかポリゴンの数より大きい場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

multipolygon
index で示されたポリゴンを戻す複数ポリゴンを表す、ST_MultiPolygon タイプの値。

index *multipolygon* から戻される *n* 番目のポリゴンを示す、INTEGER タイプの値。

戻りタイプ

db2gse.ST_Polygon

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_PolygonN の使用法を示しています。

```
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
```

結果:

```
ID        SECOND_INDEX
--------- ------------------------------------------------------------
        1 POLYGON (( 8.00000000 24.00000000, 9.00000000 25.00000000,
                     1.00000000 28.00000000, 8.00000000 24.00000000))
```

ST_Relate

ST_Relate は、2 つの形状と DE-9IM (Dimensionally Extended 9 Intersection Model) マトリックスを入力パラメーターとし、与えられた形状がマトリックスで示された条件を満たす場合に 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

与えられた形状のいずれかが NULL 値または空の場合は、NULL 値が戻されます。

2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry1
geometry2 に対してテストされる形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2
geometry1 に対してテストされる形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

matrix *geometry1* と *geometry2* のテストに使用される、DE-9IM マトリックスを表す、CHAR(9) の値。

戻りタイプ

INTEGER

例

次のコードは、次の 2 つの離れたポリゴンを作成します。次に、ST_Relate 関数を使用して、この 2 つのポリゴン間にあるいくつかのリレーションシップを判別します。例えば、2 つのポリゴンが重なり合うかどうかといった関係です。

```
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
```

結果:

```
Overlaps    Contains    Within      Intersects  Equals
----------- ----------- ----------- ----------- -----------
          1           0           0           1           0
```

ST_RemovePoint

ST_RemovePoint は曲線とポイントを入力パラメーターとし、指定されたポイントと等しいポイントをすべて与えられた曲線から除去して戻します。与えられた曲線が Z または M 座標を持つ場合、ポイントも Z または M 座標を持つ必要があります。結果の形状は、与えられた形状の空間参照系で表現されます。

与えられた曲線が空の場合は、空の曲線が戻されます。与えられた曲線が NULL の場合、または与えられたポイントが NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

curve *point* を除去する曲線を表す、ST_Curve タイプまたはそのサブタイプの 1 つの値。

point *curve* から除去するポイントを示す、ST_Point タイプの値。

戻りタイプ

db2gse.ST_Curve

例

例 1

次の例は、SAMPLE_LINES 表に 2 つの折れ線を追加します。これらの折れ線は、以下の例で使用されます。

```
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))
```

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

例 2

次の例は、ID = 1 を持つ折れ線からポイント (5, 5) を除去します。このポイントは折れ線内に 2 回発生します。したがって、両方とも除去されます。

```
SELECT CAST(ST_AsText (ST_RemovePoint (line, ST_Point(5, 5) ) )
  AS VARCHAR(120) ) RESULT
  FROM sample_lines
  WHERE id = 1
```

結果:

```
RESULT
------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 0.00000000 0.00000000,
             10.00000000 0.00000000, 0.00000000 10.00000000)
```

例 3

次の例は、ID = 2 を持つ折れ線からポイント (5, 5, 5) を除去します。このポイントは 1 つだけなので、そこだけが除去されます。

```
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
```

結果:

```
RESULT
------------------------------------------------------------------------
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 10.00000000 10.00000000
               6.00000000,  5.00000000 5.00000000 7.00000000, 0.00000000
               0.00000000 8.00000000)
```

ST_SrsId、ST_SRID

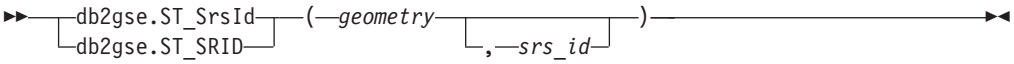
ST_SrsId (または ST_SRID) は、形状および (オプションとして) 空間参照系 ID を入力パラメーターとしてとります。何を戻すかは、入力パラメーターに何を指定したかにより異なります。

- 空間参照系 ID を指定した場合は、形状の空間参照系を指定された空間参照系に変更した形状が戻されます。形状のトランスフォーメーションは行われません。
- 入力パラメーターに空間参照系 ID を指定しないと、与えられた形状の現行の空間参照系 ID が戻されます。

与えられた形状が NULL の場合は NULL が戻されます。

これらの関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
空間参照系 ID をセットする、またはその ID を戻す形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

srs_id 結果の形状に使用される空間参照系を識別する、INTEGER タイプの値。

重要: このパラメーターを指定した場合、形状はトランスフォームされませんが、形状の空間参照系は指定された空間参照系に変更されて戻されます。新しい空間参照系に変更した結果、データが壊れる場合があります。トランスフォーメーションには、この関数ではなく ST_Transform を使用してください。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ

- srs_id が指定されていない場合は、INTEGER
- srs_id が指定されている場合は、db2gse.ST_Geometry

例

2 つの異なる空間参照系に 2 つのポイントを作成します。それぞれのポイントに関係付けられた空間参照系の ID は、ST_SrsId 関数を使用して知ることができます。

```
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
```

結果:

```
ID          SRSID
----------- -----------
          1           0
          2           1
```

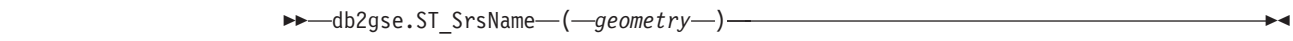
ST_SrsName

ST_SrsName は形状を入力パラメーターとし、与えられた形状を表現する空間参照系の名前を戻します。

与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
空間参照系の名前を戻す形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

VARCHAR(128)

例

異なる空間参照系に 2 つのポイントを作成します。ST_SrsName 関数を使用して、それぞれのポイントに関連付けられた空間参照系の名前を見つけます。

```
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
```

結果:

```
ID          SRSNAME
----------- ------------------------------
          1 DEFAULT_SRS
          2 NAD83_SRS_1
```

ST_StartPoint

ST_StartPoint は曲線を入力パラメーターとし、その曲線の最初のポイントを戻します。結果のポイントは、与えられた曲線の空間参照系で表現されます。この結果は、関数呼び出し ST_PointN(*curve*, 1) と同等です。

与えられた曲線が NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

curve 最初のポイントを戻す形状を表す、ST_Curve タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

db2gse.ST_Point

例

次の例は、SAMPLE_LINES 表に 2 つの折れ線を追加します。最初の折れ線は X 座標と Y 座標を持ちます。2 番目の折れ線は X 、Y、および Z 座標を持ちます。ST_StartPoint 関数を使用して、各折れ線の最初のポイントを戻します。

```
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
```

結果:

```
ID          START_POINT
----------- ----------------------------------------
          1 POINT ( 10.00000000 10.00000000)
          2 POINT Z ( 0.00000000 0.00000000 4.00000000)
```

ST_SymDifference

ST_SymDifference は 2 つの形状を入力パラメーターとし、与えられた 2 つの形状の対称差である形状を戻します。対称差 (symmetrical difference) とは、与えられた 2 つの形状の交差していない部分です。結果の形状は、1 番目の形状の空間参照系で表現されます。戻される形状のディメンションは、入力された形状と同じになります。2 つの形状は、同じディメンションの形状でなければなりません。

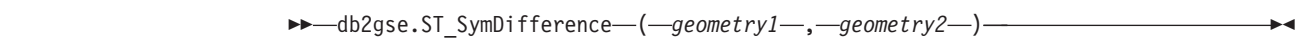
非測地データの場合、2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。測地データの場合は、両方の形状が同じ測地参照系 (SRS) で表現される必要があります。

形状が等しい場合、ST_Point タイプの空の形状が戻されます。いずれかの形状が NULL の場合は NULL が戻されます。

結果の形状は、最適な空間データ・タイプで表現されます。ポイント、折れ線、またはポリゴンとして表現できる場合は、これらのタイプの 1 つが使用されます。それ以外の場合、複数ポイント、複数折れ線、または複数ポリゴンのタイプが使用されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry1

geometry2 との対称差を計算する、1 番目の形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2

geometry1 との対称差を計算する、2 番目の形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

測地データの制約事項

- 両方の形状が測地で、しかも同じ測地 SRS で表現される必要があります。
- ST_SymDifference がサポートするデータ・タイプは、 ST_Point、 ST_Polygon、 ST_MultiPoint、 ST_MultiPolygon のみです。

戻りタイプ

db2gse.ST_Geometry

例

例 1

この例は、ST_SymDifference 関数の使用法を示しています。形状を SAMPLE_GEOMS 表に保管します。

```
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));
```

次の例では、結果は読みやすいように再フォーマットされています。結果は、ユーザーのディスプレイによって異なります。

例 2

この例は、ST_SymDifference を使用して、SAMPLE_GEOMS 表にある 2 つの結合していないポリゴンの対称差を戻します。

```
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
```

結果:

```
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)))
```

例 3

この例は、ST_SymDifference を使用して、SAMPLE_GEOMS 表にある 2 つの交差するポリゴンの対称差を戻します。

```
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
```

結果:

```
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)))
```

例 4

この例は、ST_SymDifference を使用して、SAMPLE_GEOMS 表にある 2 つの交差する折れ線の対称差を戻します。

```
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
```

結果:

```
ID   ID  SYM_DIFF
---- --- ------------------------------------------------------------------------
   4   5 MULTILINESTRING (( 70.00000000 70.00000000, 75.00000000 75.00000000),
               ( 80.00000000 80.00000000, 90.00000000 90.00000000))
```

ST_ToGeomColl

ST_ToGeomColl は形状を入力パラメーターとし、これを形状の集合に変換します。結果である形状の集合は、与えられた形状の空間参照系で表現されます。

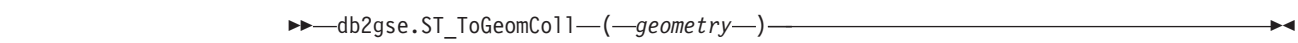
指定された形状が空の場合、どのタイプにもなり得ます。ただしこの場合、必要に応じて ST_Multipoint、ST_MultiLineString、または ST_MultiPolygon に変換されます。

指定された形状が空でない場合は、ST_Point、ST_LineString、または ST_Polygon のタイプである必要があります。これらはそれぞれ、
ST_Multipoint、ST_MultiLineString、または ST_MultiPolygon に変換されます。

与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
形状の集合に変換される形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

db2gse.ST_GeomCollection

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_ToGeomColl 関数の使用法を示しています。

```
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))
```

次の SELECT ステートメントは、ST_ToGeomColl 関数を使用して、形状をその対応する形状集合サブタイプとして戻します。

```
SELECT id, CAST( ST_AsText( ST_ToGeomColl(geometry) )
    AS VARCHAR(120) ) GEOM_COLL
FROM sample_geometries
```

結果:

```
ID          GEOM_COLL
----------- ----------------------------------------------------
          1 MULTIPOLYGON ((( 3.00000000 3.00000000, 5.00000000
                             3.00000000, 4.00000000 6.00000000,
                             3.00000000 3.00000000)))
          2 MULTIPOINT ( 1.00000000 2.00000000)
```

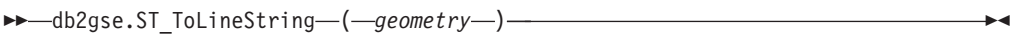
ST_ToLineString

ST_ToLineString は形状を入力パラメーターとし、これを折れ線に変換します。結果の折れ線は、与えられた形状の空間参照系で表現されます。

与える形状は空または折れ線である必要があります。与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry

折れ線に変換される形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

形状は、空であるかまたは折れ線の場合、折れ線に変換することができます。変換を実行できない場合、例外条件が戻されます (SQLSTATE 38SUD)。

戻りタイプ

db2gse.ST_LineString

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_ToLineString 関数の使用法を示します。

```
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) )
```

次の SELECT ステートメントは、ST_ToLineString 関数を使用して、ST_Geometry の静的タイプから ST_LineString に変換された折れ線を戻します。

```
SELECT CAST( ST_AsText( ST_ToLineString(geometry) )
     AS VARCHAR(130) ) LINES
FROM sample_geometries
```

結果:

```
LINES
------------------------------------------------------------------
LINESTRING Z ( 0.00000000 10.00000000 1.00000000, 0.00000000
               0.00000000 3.00000000, 10.00000000 0.00000000
               5.00000000)
LINESTRING EMPTY
LINESTRING EMPTY
```

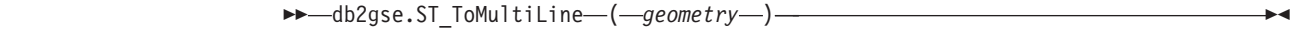
ST_ToMultiLine

ST_ToMultiLine は形状を入力パラメーターとし、これを複数折れ線に変換します。結果の複数折れ線は、与えられた形状の空間参照系で表現されます。

与える形状は空、複数折れ線、または折れ線である必要があります。与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry

複数折れ線に変換される形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

形状が、空、折れ線、または複数折れ線の場合、複数折れ線に変換することができます。変換を実行できない場合、例外条件が戻されます (SQLSTATE 38SUD)。

戻りタイプ

db2gse.ST_MultiLineString

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_ToMultiLine 関数の使用法を示しています。

```
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) )
```

次の SELECT ステートメントは、ST_ToMultiLine 関数を使用して、ST_Geometry の静的タイプから ST_MultiLineString に変換された複数折れ線を戻します。

```
SELECT CAST( ST_AsText( ST_ToMultiLine(geometry) )
  AS VARCHAR(130) ) LINES
  FROM sample_geometries
```

結果:

```
LINES
-----------------------------------------------------
MULTILINESTRING Z ( 0.00000000 10.00000000 1.00000000,
                    0.00000000 0.00000000 3.00000000,
                   10.00000000 0.00000000 5.00000000)
MULTILINESTRING EMPTY
MULTILINESTRING EMPTY
```

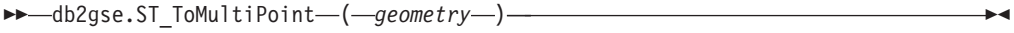
ST_ToMultiPoint

ST_ToMultiPoint は形状を入力パラメーターとし、これを複数ポイントに変換します。結果の複数ポイントは、与えられた形状の空間参照系で表現されます。

与える形状は空、ポイント、または複数ポイントである必要があります。与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
複数ポイントに変換される形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

形状は、空であるか、ポイントまたは複数ポイントの場合、複数ポイントに変換することができます。変換を実行できない場合、例外条件が戻されます (SQLSTATE 38SUD)。

戻りタイプ

db2gse.ST_MultiPoint

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_ToMultiPoint 関数の使用法を示しています。

```
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) )
```

次の SELECT ステートメントは、ST_ToMultiPoint 関数を使用して、ST_Geometry の静的タイプから ST_MultiPoint に変換された複数ポイントを戻します。

```
SELECT CAST( ST_AsText( ST_ToMultiPoint(geometry))
  AS VARCHAR(62) ) MULTIPOINTS
FROM sample_geometries
```

結果:

```
MULTIPOINTS
------------------------------------------------------------
MULTIPOINT ( 0.00000000 0.00000000, 0.00000000 4.00000000)
MULTIPOINT ( 30.00000000 40.00000000)
MULTIPOINT EMPTY
```

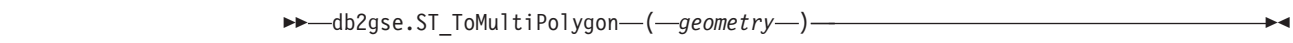
ST_ToMultiPolygon

ST_ToMultiPolygon は形状を入力パラメーターとし、これを複数ポリゴンに変換します。結果の複数ポリゴンは、与えられた形状の空間参照系で表現されます。

与える形状は空、ポリゴン、または複数ポリゴンである必要があります。与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry
複数ポリゴンに変換される形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

形状は、空であるか、ポリゴンまたは複数ポリゴンの場合、複数ポリゴンに変換することができます。変換を実行できない場合、例外条件が戻されます (SQLSTATE 38SUD)。

戻りタイプ

db2gse.ST_MultiPolygon

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は複数の形状を作成し、ST_ToMultiPolygon を使用して複数ポリゴンを戻します。

```
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))
```

次の SELECT ステートメントは、ST_ToMultiPolygon 関数を使用して、ST_Geometry の静的タイプから ST_MultiPolygon に変換された複数ポリゴンを戻します。

```
SELECT CAST( ST_AsText( ST_ToMultiPolygon(geometry) )
  AS VARCHAR(130) ) POLYGONS
  FROM sample_geometries
```

結果:

```
POLYGONS
------------------------------------------------------------------------
MULTIPOLYGON (( 0.00000000 0.00000000, 5.00000000 0.00000000,
                5.00000000 4.00000000, 0.00000000 4.00000000,
                 0.00000000 0.00000000))

MULTIPOLYGON EMPTY

MULTIPOLYGON EMPTY
```

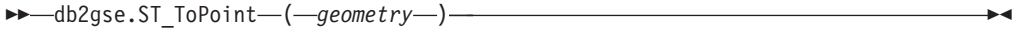
ST_ToPoint

ST_ToPoint は形状を入力パラメーターとし、これをポイントに変換します。結果のポイントは、与えられた形状の空間参照系で表現されます。

与える形状は空またはポイントである必要があります。与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry

ポイントに変換される形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

形状は、空であるかまたはポイントの場合、ポイントに変換することができます。変換を実行できない場合、例外条件が戻されます (SQLSTATE 38SUD)。

戻りタイプ

db2gse.ST_Point

例

この例は SAMPLE_GEOMETRIES に 3 つの形状を作成し、それぞれをポイントに変換します。

```
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) )
```

次の SELECT ステートメントは、ST_ToPoint 関数を使用して、ST_Geometry の静的タイプから ST_Point に変換されたポイントを戻します。

```
SELECT CAST( ST_AsText( ST_ToPoint(geometry) ) AS VARCHAR(35) ) POINTS
  FROM sample_geometries
```

結果:

```
POINTS
---------------------------------
POINT ( 30.00000000 40.00000000)
POINT EMPTY
POINT EMPTY
```

ST_ToPolygon

ST_ToPolygon は形状を入力パラメーターとし、これをポリゴンに変換します。結果のポリゴンは、与えられた形状の空間参照系で表現されます。

与える形状は空またはポリゴンである必要があります。与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry

ポリゴンに変換される形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

形状は、空であるかまたはポリゴンの場合、ポリゴンに変換することができます。変換を実行できない場合、例外条件が戻されます (SQLSTATE 38SUD)。

戻りタイプ

db2gse.ST_Polygon

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は SAMPLE_GEOMETRIES に 3 つの形状を作成し、それぞれをポリゴンに変換します。

```
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) )
```

次の SELECT ステートメントは、ST_ToPolygon 関数を使用して、ST_Geometry の静的タイプから ST_Polygon に変換されたポリゴンを戻します。

```
SELECT CAST( ST_AsText( ST_ToPolygon(geometry) ) AS VARCHAR(130) ) POLYGONS
  FROM sample_geometries
```

結果:

```
POLYGONS
--------------------------------------------------------------------------------
POLYGON (( 0.00000000 0.00000000, 5.00000000 0.00000000,
           5.00000000 4.00000000,0.00000000 4.00000000,
            0.00000000 0.00000000))

POLYGON EMPTY

POLYGON EMPTY
```

ST_Touches

ST_Touches は 2 つの形状を入力パラメーターとし、与えられた形状が空間的に接触する場合は 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

「2 つの形状が接触する」とは、両方の形状の内部は交差していないが、一方の形状の境界が、他方の形状の境界または内部と交差している場合です。

2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。

与えられた形状の両方がポイントまたは複数ポイントの場合、または与えられた形状のいずれかが NULL または空の場合は、NULL が戻されます。

構文

パラメーター

geometry1

geometry2 との接触をテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2

geometry1 との接触をテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

戻りタイプ

INTEGER

例

いくつかの形状を SAMPLE_GEOMS 表に追加します。次に ST_Touches 関数を使用して、どの形状がお互いに接触するかを判別します。

```
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
```

結果:

```
     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
```

ST_Transform

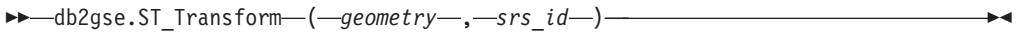
ST_Transform は、形状および空間参照系 ID を入力パラメーターとし、指定された空間参照系で表現されるようにその形状をトランスフォームします。異なる座標システムの間で投影および変換が行われ、形状の座標はそれに従って調整されます。

形状を指定された空間参照系に変換できるのは、形状の現行の空間参照系が、指定された空間参照系と同じ地理座標システムに基づく場合のみです。形状の現行の空間参照系または指定された空間参照系のいずれかが、投影座標システムに基づく場合、投影されたものの基礎にある地理座標システムを判別するため、逆の投影が行われます。

与えられた形状が NULL の場合は NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometry

srs_id で示された空間参照系にトランスフォームされる形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

srs_id 結果の形状の空間参照系を識別する、INTEGER タイプの値。

geometry の現行の空間参照系が、*srs_id* で示された空間参照系と互換性がないため、指定された空間参照系へのトランスフォーメーションができない場合は、例外条件 (SQLSTATE 38SUC) が起こります。

srs_id が、カタログ・ビュー DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS にリストされた空間参照系でない場合は、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ

db2gse.ST_Geometry

例

例 1

次の例は、ST_Transform を使用して、ある空間参照系から別の空間参照系に形状を変換しています。

最初に、db2se 呼び出しを使用して、ID 3 を持つ州平野の空間参照系を作成します。

```
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
```

次に、以下の表にポイントを追加します。

- 空間参照系を使用する、state plane 座標システムの SAMPLE_POINTS_SP 表。
- 緯度と経度で指定した座標を使用する SAMPLE_POINTS_LL 表。

```
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) )
```

次に、ST_Transform 関数を使用して形状を変換します。

例 2

この例は、緯度と経度の座標のポイントを state plane 座標に変換します。

```
SELECT id, CAST( ST_AsText( ST_Transform( geometry, 3) )
    AS VARCHAR(100) ) STATE_PLANE
  FROM sample_points_ll
```

結果:

```
ID          STATE_PLANE
----------- ----------------------------------------
      12457 POINT ( 567176.00000000 1166411.00000000)
      12477 POINT ( 637948.00000000 1177640.00000000)
```

例 3

この例は、state plane 座標のポイントを、緯度と経度の座標に変換します。

```
SELECT id, CAST( ST_AsText( ST_Transform( geometry, 1) )
    AS VARCHAR(100) ) LAT_LONG
  FROM sample_points_sp
```

結果:

```
ID          LAT_LONG
----------- ----------------------------------------------------------------
      12457 POINT ( -74.22371500 42.03498800)
      12477 POINT ( -73.96293100 42.06488000)
```

ST_Union

ST_Union は 2 つの形状を入力パラメーターとし、与えられた形状の和である形状を戻します。結果の形状は、1 番目の形状の空間参照系で表現されます。

2 つの形状は、同じディメンションの形状でなければなりません。与えられた 2 つの形状のいずれかが NULL の場合は、NULL が戻されます。

非測地データの場合、2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。測地データの場合は、両方の形状が同じ測地参照系 (SRS) で表現される必要があります。

結果の形状は、最適な空間データ・タイプで表現されます。ポイント、折れ線、またはポリゴンとして表現できる場合は、これらのタイプの 1 つが使用されます。それ以外の場合、複数ポイント、複数折れ線、または複数ポリゴンのタイプが使用されます。

この関数はメソッドとして呼び出すこともできます。

構文

パラメーター

geometry1
geometry2 と結合される、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2
geometry1 と結合される形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

測地データの制限事項: 両方の形状が測地で、しかも同じ測地 SRS で表現される必要があります。

戻りタイプ

db2gse.ST_Geometry

例

例 1

以下の SQL ステートメントは、SAMPLE_GEOM テーブルを作成し、データを設定するものです。

```
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))
```

次の例では、結果は読みやすいように再フォーマットされています。結果は、ユーザーのディスプレイによって異なります。

例 2

この例は 2 つの分離したポリゴンの和を作成します。

```
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
```

結果:

```
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)))
```

例 3

この例は 2 つの交差するポリゴンの和を作成します。

```
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
```

結果:

```
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))
```

例 4

2 つの折れ線の和を作成します。

```
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
```

結果:

```
ID    ID    UNION
----- ----- ------------------------------------------------------------
    4     5 MULTILINESTRING (( 70.00000000 70.00000000, 80.00000000 80.00000000),
            ( 80.00000000 80.00000000, 100.00000000 70.00000000))
```

ST_Within

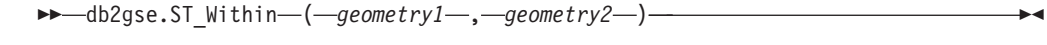
ST_Within は 2 つの形状を入力パラメーターとし、1 番目の形状が完全に 2 番目の中にある場合に 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

与えられた形状のいずれかが NULL 値または空の場合は、NULL 値が戻されます。

非測地データの場合、2 番目の形状が 1 番目の形状と同じ空間参照系で表現されない場合は、他の空間参照系に変換されます。測地データの場合は、両方の形状が同じ測地参照系 (SRS) で表現される必要があります。

ST_Within は、パラメーターを逆にして ST_Contains を実行した場合と論理的に同じ操作を実行します。

構文



パラメーター

geometry1
全体が *geometry2* の中にあるかどうかをテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

geometry2
全体が *geometry1* の中にあるかどうかをテストする形状を表す、ST_Geometry タイプまたはそのサブタイプの 1 つの値。

測地データの制限事項: 両方の形状が測地で、しかも同じ測地 SRS で表現される必要があります。

戻りタイプ

INTEGER

例

例 1

この例は、ST_Within 関数の使用法を示しています。形状を作成し、3 つの表 SAMPLE_POINTS、SAMPLE_LINES、および SAMPLE_POLYGONS に挿入します。

```
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) )
```

例 2

この例は、SAMPLE_POINTS 表から、SAMPLE_POLYGONS 表のポリゴンの中にあるポイントを見つけます。

```
SELECT a.id POINT_ID_WITHIN_POLYGONS
  FROM sample_points a, sample_polygons b
  WHERE ST_Within( b.geometry, a.geometry) = 0
```

結果:

```
POINT_ID_WITHIN_POLYGONS
----------------------
                     2
```

例 3

この例は、SAMPLE_LINES 表から、SAMPLE_POLYGONS 表のポリゴンの中にある折れ線を見つけます。

```
SELECT a.id LINE_ID_WITHIN_POLYGONS
  FROM sample_lines a, sample_polygons b
  WHERE ST_Within( b.geometry, a.geometry) = 0
```

結果:

```
LINE_ID_WITHIN_POLYGONS
----------------------
                     1
```

ST_WKBToSQL

ST_WKBToSQL は、形状の事前割り当てバイナリー表記を取り、これに対応する形状を戻します。結果の形状には、ID 0 (ゼロ) を持つ空間参照系が使用されます。

与えられた、事前割り当てバイナリー表記が NULL の場合は、NULL が戻されます。

ST_WKBToSQL(*wkb*) は、ST_Geometry(*wkb*,0) と同じ結果を得ることができます。ST_WKBToSQL を使用するよりも ST_Geometry 関数を使用することをお勧めします。その理由は、ST_Geometry は事前割り当てバイナリー表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文

パラメーター

wkb 結果の形状の事前割り当てバイナリー表記が入る、タイプ BLOB(2G) の値。

戻りタイプ

db2gse.ST_Geometry

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_WKBToSQL 関数の使用法を示しています。最初に、形状を SAMPLE_GEOMETRIES 表の GEOMETRY 列に保管します。次に、UPDATE ステートメントで ST_AsBinary 関数を使用して、この形状の事前割り当てバイナリー表記を WKB 列に保管します。最後に ST_WKBToSQL 関数を使用して、WKB 列の形状の座標を戻します。

```
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
```

次の SELECT ステートメントを使用して、WKB 列にある形状を調べます。

```
SELECT id, CAST( ST_AsText( ST_WKBToSQL(wkb) ) AS VARCHAR(120) ) GEOMETRIES
  FROM sample_geometries
```

結果:

```
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))
```

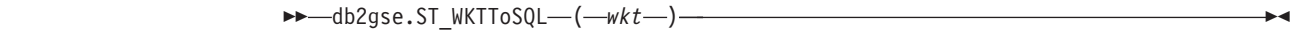
ST_WKTToSQL

ST_WKTToSQL は、形状の事前割り当てテキスト表記を取り、対応する形状を戻します。結果の形状には、ID 0 (ゼロ) を持つ空間参照系が使用されます。

指定された事前割り当てテキスト表記が NULL の場合は、NULL が戻されます。

ST_WKTToSQL(*wkt*) は、ST_Geometry(*wkt*,0) と同じ結果を得ることができます。ST_WKTToSQL を使用するよりも ST_Geometry 関数を使用することをお勧めします。その理由は、ST_Geometry は事前割り当てテキスト表記だけでなく、追加の入力フォーマットを処理できる柔軟性があるためです。

構文



パラメーター

wkt 結果の形状の事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。

戻りタイプ

db2gse.ST_Geometry

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、ST_WKTToSQL により、形状の事前割り当てテキスト表記を使用して形状を作成、挿入する方法を示しています。

```
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))' ) )
```

この SELECT ステートメントは、挿入された形状を戻します。

```
SELECT id, CAST( ST_AsText(geometry) AS VARCHAR(120) ) GEOMETRIES
  FROM sample_geometries
```

結果:

```
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))
```

ST_X

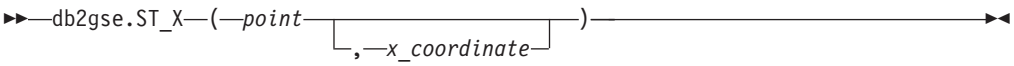
ST_X は次のいずれかを行います。

- ポイントを入力パラメーターとし、その X 座標を戻す
- ポイントと X 座標を入力とし、その X 座標を与えられた値に設定して、そのポイント自体を戻す

与えられたポイントが NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

point　X 座標を戻すかまたは変更したい値 (タイプ ST_Point)。

x_coordinate
point の新しい X 座標を表す DOUBLE タイプの値。

戻りタイプ

- *x_coordinate* が指定されていない場合は、DOUBLE
- *x_coordinate* が指定されている場合は、db2gse.ST_Point

例

例 1

この例は、ST_X 関数の使用法を示しています。形状を作成し、SAMPLE_POINTS 表に挿入します。

```
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) )
```

例 2

この例は、表内のポイントの X 座標を見つけます。

```
SELECT id, ST_X (geometry) X_COORD
  FROM sample_points
```

結果:

```
ID          X_COORD
----------- ----------------------
          1 +2.00000000000000E+000
          2 +4.00000000000000E+000
          3 +3.00000000000000E+000
```

例 3

この例は、X 座標が 40 にセットされているポイントを戻します。

```
SELECT id, CAST( ST_AsText( ST_X (geometry, 40)) AS VARCHAR(60) )
  X_40
  FROM sample_points
  WHERE id=3
```

結果:

```
ID          X_40
------ ------------------------------------------------------------
     3 POINT ZM ( 40.00000000 8.00000000 23.00000000 7.00000000)
```

ST_Y

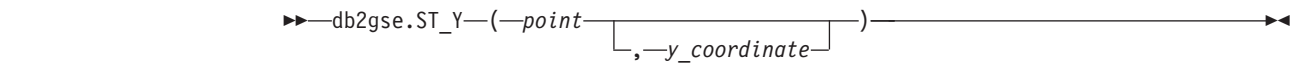
ST_Y は次のいずれかを行います。

- ポイントを入力パラメーターとし、その Y 座標を戻す
- ポイントと Y 座標を入力とし、そのポイント自体および、与えられた値にセットされた Y 座標を戻す

与えられたポイントが NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

point Y 座標を戻すかまたは変更したい値 (タイプ ST_Point)。

y_coordinate
point の新しい Y 座標を表す DOUBLE タイプの値。

戻りタイプ

- *y_coordinate* が指定されていない場合は、DOUBLE
- *y_coordinate* が指定されている場合は、db2gse.ST_Point

例

例 1

この例は、ST_Y 関数の使用法を示しています。形状を作成し、SAMPLE_POINTS 表に挿入します。

```
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) )
```

例 2

この例は、表内のポイントの Y 座標を見つけます。

```
SELECT id, ST_Y (geometry) Y_COORD
  FROM sample_points
```

結果:

```
ID          Y_COORD
----------- ----------------------
          1 +3.00000000000000E+000
          2 +5.00000000000000E+000
          3 +8.00000000000000E+000
```

例 3

この例は、Y 座標が 40 のポイントを戻します。

```
SELECT id, CAST( ST_AsText( ST_Y (geometry, 40)) AS VARCHAR(60) )
  Y_40
  FROM sample_points
  WHERE id=3
```

結果:

```
ID         Y_40
-------- ------------------------------------------------------
     3 POINT ZM ( 3.00000000 40.00000000 23.00000000 7.00000000)
```

ST_Z

ST_Z は次のいずれかを行います。

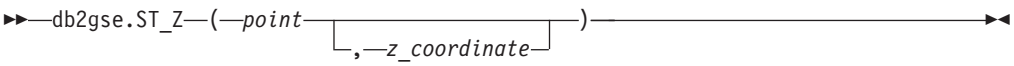
- ポイントを入力パラメーターとし、その Z 座標を戻す
- ポイントと Z 座標を入力パラメーターとし、ポイント自体と、与えられた値にセットされた Z 座標を戻す (指定されたポイントに Z 座標が存在しない場合でも)

指定された Z 座標が NULL の場合、ポイントの Z 座標は除去されます。

指定されたポイントが NULL または空の場合は、NULL が戻されます。

この関数はメソッドとして呼び出すこともできます。

構文



パラメーター

point Z 座標を戻すかまたは変更したい値 (タイプ ST_Point)。

z_coordinate

point の新規 Z 座標を表す DOUBLE タイプの値。

z_coordinate が NULL の場合、Z 座標は *point* から除去されます。

戻りタイプ

- *z_coordinate* が指定されていない場合は、DOUBLE
- *z_coordinate* が指定されている場合は、db2gse.ST_Point

例

例 1

この例は、ST_Z 関数の使用法を示しています。形状を作成し、SAMPLE_POINTS 表に挿入します。

```
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) )
```

例 2

この例は、表内のポイントの Z 座標を見つけます。

```
SELECT id, ST_Z (geometry) Z_COORD
  FROM sample_points
```

結果:

```
ID          Z_COORD
----------- ----------------------
          1 +3.20000000000000E+001
          2 +2.00000000000000E+001
          3 +2.30000000000000E+001
```

例 3

この例は、Z 座標が 40 のポイントを戻します。

```
SELECT id, CAST( ST_AsText( ST_Z (geometry, 40)) AS VARCHAR(60) )
  Z_40
  FROM sample_points
  WHERE id=3
```

結果:

```
ID          Z_40
------ ------------------------------------------------------
     3 POINT ZM ( 3.00000000 8.00000000 40.00000000 7.00000000)
```

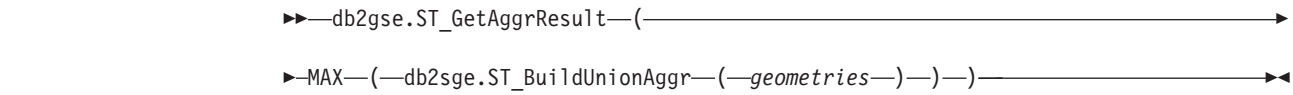
和集約

和集約は、ST_BuildUnionAggr と ST_GetAggrResult の関数を組み合わせたものです。この組み合わせは、表内の形状の列を和集約として集約し、1 つの形状にします。

和として結合する形状のすべてが NULL の場合は、NULL が戻されます。和として結合する形状のそれぞれが NULL または空の場合は、ST_Point タイプの空の形状が戻されます。

ST_BuildUnionAggr 関数はメソッドとして呼び出すこともできます。

構文



パラメーター

geometries

ST_Geometry タイプまたはそのサブタイプの 1 つの、表内の列であり、和として結合するすべての形状を表す列。

戻りタイプ

db2gse.ST_Geometry

制約事項

以下のいずれかに該当する場合は、表の空間列の和集約を作成できません。

- データベース・パーティション・フィーチャー (DPF) 環境内。
- SELECT で GROUP BY 節を使用した場合
- DB2 集約関数 MAX 以外の関数を使用した場合

例

次の例では、結果の行は読みやすいように再フォーマットされています。結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

この例は、和集約を使用して、ポイントのセットを複数ポイントに結合する方法を示しています。いくつかのポイントを SAMPLE_POINTS 表に追加します。ST_GetAggrResult および ST_BuildUnionAggr 関数を使用して、ポイントの和集合を作成します。

```
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
```

結果:

```
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)
```

第 24 章 トランスフォーム・グループ

トランスフォーム・グループ

Spatial Extender は、DB2 サーバーとクライアント・アプリケーションの間で形状を転送するときに使用される 4 つのトランスフォーム・グループを提供しています。これらのトランスフォーム・グループは、以下のデータ交換フォーマットに対応しています。

- 事前割り当てテキスト (WKT) 表記
- 事前割り当てバイナリー (WKB) 表記
- ESRI 形状表記
- Geography Markup Language (GML)

空間列を含んでいる表からデータを検索する場合、空間列からのデータは、トランスフォームされたデータをバイナリー形式またはテキスト形式のどちらで表示するよう指定したかに応じて、 CLOB(2G) または BLOB(2G) のいずれかにトランスフォームされます。データベースに空間データを転送するときに、トランスフォーム・グループを使用することもできます。

データを転送するときに使用するトランスフォーム・グループを選択するには、SET CURRENT DEFAULT TRANSFORM GROUP ステートメントを使用して、DB2 特殊レジスター CURRENT DEFAULT TRANSFORM GROUP を変更する必要があります。 DB2 は、特殊レジスターの値を使用して、必要な変換を行うために呼び出すべきトランスフォーム関数を決定します。

トランスフォーム・グループにより、アプリケーション・プログラミングが簡単になります。 SQL ステートメントで変換関数を明示的に使用する代わりに、トランスフォーム・グループを指定して、 DB2 にそのタスクを処理させることができます。

ST_WellKnownText トランスフォーム・グループ

ST_WellKnownText トランスフォーム・グループは、事前割り当てテキスト (WKT) 表記を使用して DB2® との間でデータを伝送するために使用することができます。

データベース・サーバーからクライアントへ値をバインドアウトする場合、ST_AsText() によって提供される同じ機能が形状を WKT 表記に変換するために使用されます。形状の事前割り当てテキスト表記をデータベース・サーバーに転送する場合、 ST_Geometry(CLOB) 関数が暗黙的に使用されて、 ST_Geometry 値への変換を行います。値を DB2 へバインドインするためにトランスフォーム・グループを使用すると、形状は、数値 ID 0 (ゼロ) の空間参照系で表現されます。

例

次の例では、結果の行は読みやすいように再フォーマットされています。 結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

© Copyright IBM Corp. 1998, 2009

例 1

次の SQL スクリプトは、明示的に ST_AsText 関数を使用することなく、ST_WellKnownText トランスフォーム・グループをどのように使用して、事前割り当てテキスト表記で形状を検索できるかを示しています。

```
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
```

結果:

```
ID   GEOM
---  ------------------------------------------------------------------
  1  LINESTRING ( 100.00000000 100.00000000, 200.00000000 100.00000000)
```

例 2

次の C コードは、タイプが CLOB の変数で、挿入されるポイント (10 10) の事前割り当てテキスト表記を含んでいるホスト変数 wkt_buffer のための明示的な ST_Geometry 関数を使用して形状を挿入するために、 ST_WellKnownText トランスフォーム・グループを使用する方法を示しています。

```
EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS CLOB(1000) wkt_buffer;
EXEC SQL END DECLARE SECTION;

// set the transform group for all subsequent SQL statements
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);

// insert point using WKT into column of type ST_Geometry
EXEC SQL
   INSERT
    INTO   transforms_sample(id, geom)
    VALUES (:id, :wkt_buffer);
```

ST_WellKnownBinary トランスフォーム・グループ

ST_WellKnownBinary トランスフォーム・グループは、事前割り当てバイナリー (WKB) 表記を使用して DB2® との間でデータを伝送するために使用することができます。

データベース・サーバーからクライアントへ値をバインドアウトする場合、ST_AsBinary() によって提供される同じ機能が形状を WKB 表記に変換するために

使用されます。形状の事前割り当てバイナリー表記をデータベース・サーバーに転送する場合、 ST_Geometry(BLOB) 関数が暗黙的に使用されて、 ST_Geometry 値への変換を行います。値を DB2 へバインドインするためにトランスフォーム・グループを使用すると、形状は、数値 ID 0 (ゼロ) の空間参照系で表現されます。

例

次の例では、結果の行は読みやすいように再フォーマットされています。 結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

例 1

次の SQL スクリプトは、明示的に ST_AsBinary 関数を使用することなく、ST_WellKnownBinary トランスフォーム・グループをどのように使用して、事前割り当てバイナリー表記で形状を検索できるかを示しています。

```
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
```

結果:

```
ID    GEOM
----   ------------------------------------------------
   1   x'0103000000010000000500000000000000000024400000
       0000000000244000000000000024400000000000003440
       0000000000003440000000000000344000000000000034
       4000000000000024400000000000002440000000000000
       2440'
```

例 2

次の C コードは、タイプ BLOB の変数で、挿入される形状の事前割り当てバイナリー表記を含んでいるホスト変数 wkb_buffer のための明示的な ST_Geometry 関数を使用して形状を挿入するために、 ST_WellKnownBinary トランスフォーム・グループを使用する方法を示しています。

```
EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS BLOB(1000) wkb_buffer;
EXEC SQL END DECLARE SECTION;

// set the transform group for all subsequent SQL statements
EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownBinary;

// initialize host variables
...
// insert geometry using WKB into column of type ST_Geometry
```

```
EXEC SQL
   INSERT
    INTO transforms_sample(id, geom)
    VALUES ( :id, :wkb_buffer );
```

ST_Shape トランスフォーム・グループ

ST_Shape トランスフォーム・グループは、ESRI 形状表記を使用して DB2® との間でデータを伝送するために使用することができます。

データベース・サーバーからクライアントへ値をバインドアウトする場合、ST_AsShape() によって提供される同じ機能が、形状をその形状表記に変換するために使用されます。形状の形状表記をデータベース・サーバーに転送する場合、ST_Geometry(BLOB) 関数が暗黙的に使用されて、ST_Geometry 値への変換を行います。値を DB2 へバインドインするためにトランスフォーム・グループを使用すると、形状は、数値 ID 0 (ゼロ) の空間参照系で表現されます。

例

次の例では、結果の行は読みやすいように再フォーマットされています。 結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

例 1

次の SQL スクリプトは、明示的な ST_AsShape 関数を使用することなく、形状をその形状表記で検索するために、ST_Shape トランスフォーム・グループを使用する方法を示しています。

```
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
```

結果:

```
ID    GEOM
----   ---------------------------------------------
   1    x'0100000000000000000003440000000000000003E40'
```

例 2

次の C コードは、タイプ BLOB の変数で、挿入される形状の形状表記を含んでいるホスト変数 shape_buffer のための明示的な ST_Geometry 関数を使用して形状を挿入するために、 ST_Shape トランスフォーム・グループを使用する方法を示しています。

```
EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS BLOB(1000) shape_buffer;
EXEC SQL END DECLARE SECTION;
```

```
// set the transform group for all subsequent SQL statements
EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// initialize host variables
...

SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// insert geometry using shape representation into column of type ST_Geometry
EXEC SQL
     INSERT
    INTO transforms_sample(id, geom)
    VALUES ( :id, :shape_buffer );
```

ST_GML トランスフォーム・グループ

ST_GML トランスフォーム・グループは、Geography Markup Language (GML) を使用して DB2® との間でデータを伝送するために使用することができます。

データベース・サーバーからクライアントへ値をバインドアウトする場合、ST_AsGML() によって提供される同じ機能が、形状をその GML 表記に変換するために使用されます。 GML 表記をデータベース・サーバーに転送する場合、ST_Geometry(CLOB) 関数が暗黙的に使用されて、 ST_Geometry 値への変換を行います。値を DB2 へバインドインするためにトランスフォーム・グループを使用すると、形状は、数値 ID 0 (ゼロ) の空間参照系で表現されます。

例

次の例では、結果の行は読みやすいように再フォーマットされています。 結果におけるスペーシングは、ユーザーのオンライン・ディスプレイによって異なります。

例 1

次の SQL スクリプトは、明示的な ST_AsGML 関数を使用することなく、形状をその GML 表記で検索するために、ST_GML トランスフォーム・グループを使用する方法を示しています。

```
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
```

結果:

```
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>
```

例 2

次の C コードは、タイプ CLOB の変数で、挿入されるポイント (20、20) の GML 表記を含んでいるホスト変数 gml_buffer のための明示的な ST_Geometry 関数を使用せずに形状を挿入するために、 ST_GML トランスフォーム・グループを使用する方法を示しています。

```
EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS CLOB(1000) gml_buffer;
EXEC SQL END DECLARE SECTION;

// set the transform group for all subsequent SQL statements
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>");

//initialize host variables
wkt_buffer.length = strlen(gml_buffer.data);

// insert point using WKT into column of type ST_Geometry
EXEC SQL
   INSERT
   INTO transforms_sample(id, geom)
   VALUES ( :id, :gml_buffer );
```

第 25 章 サポートされるデータ・フォーマット

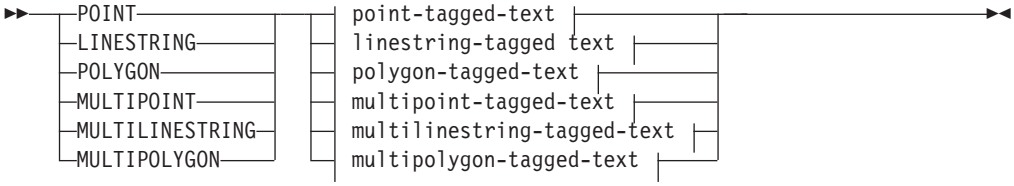
この章では、DB2 Spatial Extender で使用できる、業界標準の空間データ・フォーマットを説明します。 以下の 4 つの空間データ・フォーマットが説明されています。

- 事前割り当てテキスト (WKT) 表記
- 事前割り当てバイナリー (WKB) 表記
- 形状表記
- Geography Markup Language (GML) 表記

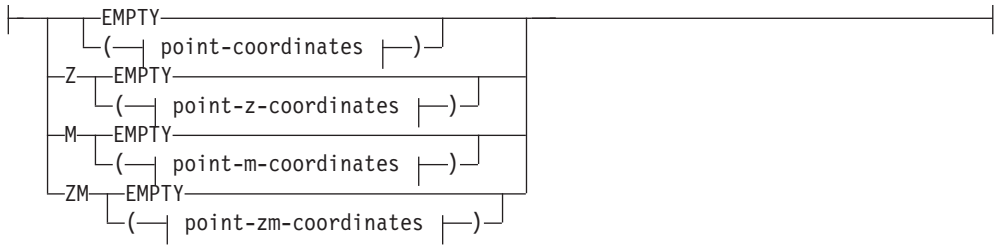
事前割り当てテキスト (WKT) 表記

OpenGIS コンソーシアムの「Simple Features for SQL」仕様に、ASCII フォーマットの形状データを交換するための事前割り当てテキスト表記が定義されています。この表記は、ISO 「SQL/MM Part: 3 Spatial」標準でも参照されています。WKT データを受け付け、作成する関数の情報については、『データ交換フォーマットの形状を変換する空間処理関数』を参照してください。

形状の事前割り当てテキスト表記は、次のように定義されます。

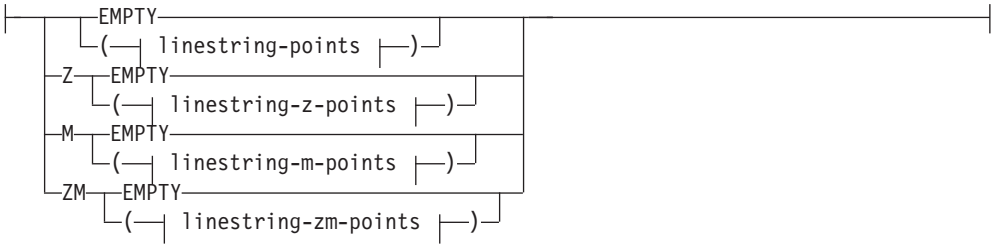


point-tagged-text:

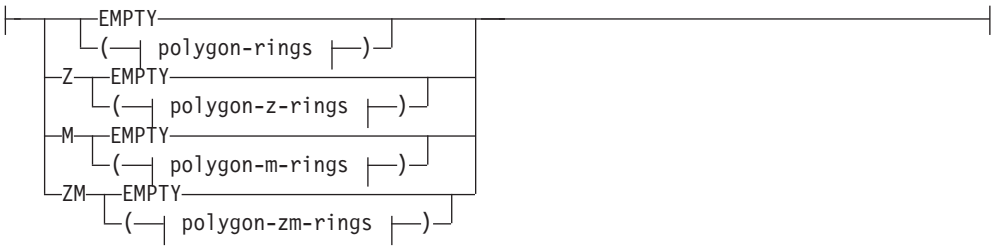


© Copyright IBM Corp. 1998, 2009

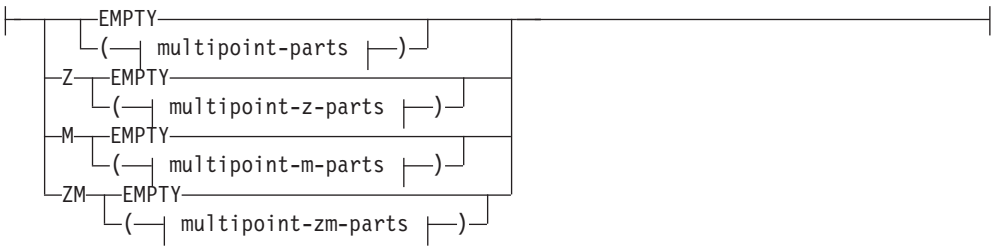
linestring-tagged-text:



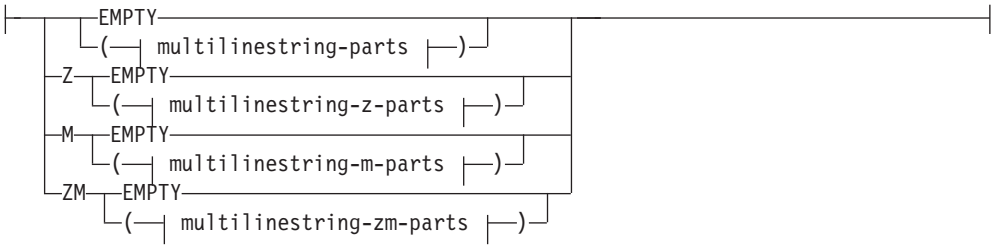
polygon-tagged-text:



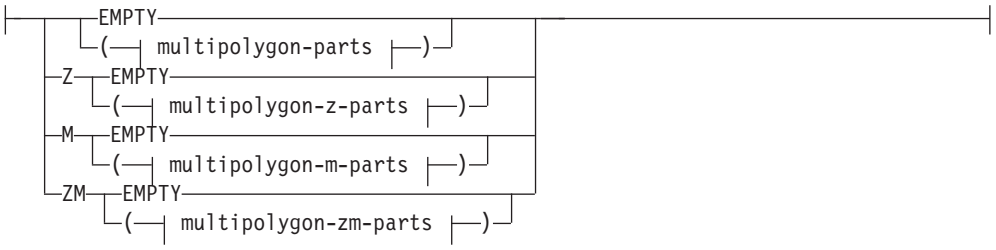
multipoint-tagged-text:



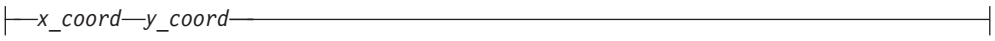
multilinestring-tagged-text:



multipolygon-tagged-text:

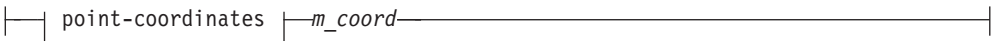


point-coordinates:

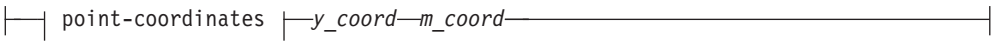


point-z-coordinates:

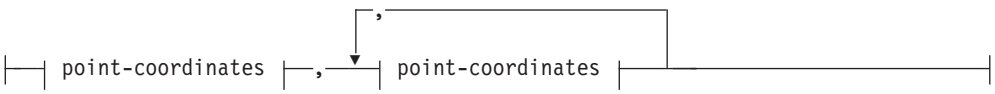
point-m-coordinates:



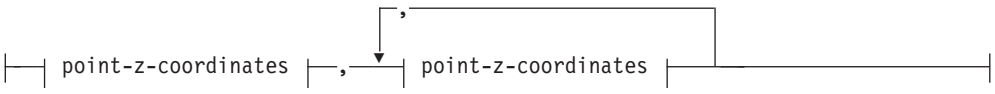
point-zm-coordinates:



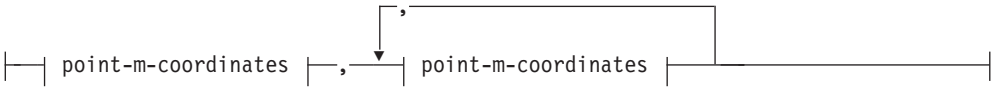
linestring-points:



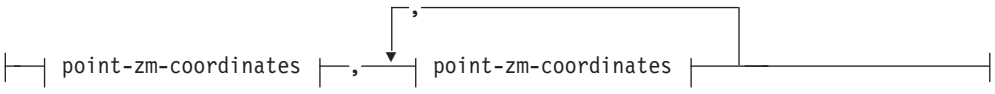
linestring-z-points:



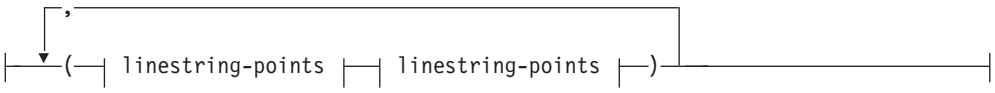
linestring-m-points:



linestring-zm-points:



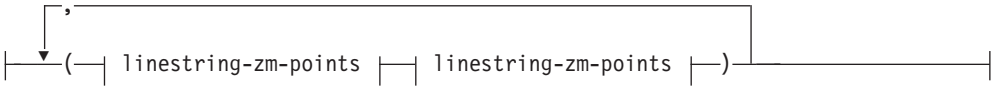
polygon-rings:



polygon-z-rings:

polygon-m-rings:

polygon-zm-rings:



multipoint-parts:

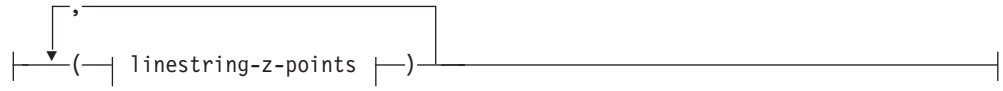
multipoint-z-parts:

multipoint-m-parts:

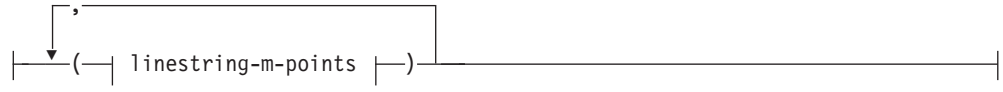
multipoint-zm-parts:

multilinestring-parts:

multilinestring-z-parts:



multilinestring-m-parts:



multilinestring-zm-parts:

multipolygon-parts:

multipolygon-z-parts:

multipolygon-m-parts:

multipolygon-zm-parts:

パラメーター

x_coord

ポイントの X 座標を表す数値 (固定、整数、または浮動小数点)。

y_coord

ポイントの Y 座標を表す数値 (固定、整数、または浮動小数点)。

z_coord

ポイントの Z 座標を表す数値 (固定、整数、または浮動小数点)。

m_coord

ポイントの M 座標 (指標) を表す数値 (固定、整数、または浮動小数点)。

形状が空の場合は、座標リストの代わりにキーワード EMPTY を指定します。EMPTY キーワードは座標リスト内に入れないでください。

次の表は、可能なテキスト表記のいくつかの例を示しています。

表 58. 形状のタイプとそのテキスト表記

形状タイプ	WKT 表記	コメント
point	POINT EMPTY	空のポイント
point	POINT (10.05 10.28)	ポイント
point	POINT Z(10.05 10.28 2.51)	Z 座標を持つポイント
point	POINT M(10.05 10.28 4.72)	M 座標を持つポイント
point	POINT ZM(10.05 10.28 2.51 4.72)	Z 座標と M 座標を持つポイント
linestring	LINESTRING EMPTY	空の折れ線
polygon	POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))	ポリゴン
multipoint	MULTIPOINT Z(10 10 2, 20 20 3)	Z 座標を持つ複数ポイント
multilinestring	MULTILINESTRING M((310 30 1, 40 30 20, 50 20 10)(10 10 0, 20 20 1))	M 座標を持つ複数折れ線
multipolygon	MULTIPOLYGON ZM(((1 1 1 1, 1 2 3 4, 2 2 5 6, 2 1 7 8, 1 1 1 1)))	Z 座標と M 座標を持つ複数ポリゴン

事前割り当てバイナリー (WKB) 表記

このセクションでは、形状の事前割り当てバイナリー表記を説明しています。

OpenGIS コンソーシアムの「Simple Features for SQL」仕様に、事前割り当てバイナリー表記が定義されています。この表記は、ISO (国際標準化機構) の「SQL/MM Part: 3 Spatial」標準でも定義されています。WKB を受け付け、作成する関数の情報については、このトピックの終わりにある関連参照セクションを参照してください。

事前割り当てバイナリー表記の基本的な構築単位は、ポイントのバイト・ストリームであり、これは 2 つのダブル値からなります。他の形状のバイト・ストリームは、既に定義済みの形状のバイト・ストリームを使用して組み立てられます。

次の例は、事前割り当てバイナリー表記の基本的な構築単位を示しています。

```
// Basic Type definitions
// byte : 1 byte
// uint32 : 32 bit unsigned integer  (4 bytes)
// double : double precision number (8 bytes)

// Building Blocks : Point, LinearRing

Point {
  double x;
  double y;
};
LinearRing   {
  uint32  numPoints;
  Point   points[numPoints];
};
enum wkbGeometryType {
  wkbPoint = 1,
  wkbLineString = 2,
  wkbPolygon = 3,
  wkbMultiPoint = 4,
  wkbMultiLineString = 5,
  wkbMultiPolygon = 6
};
enum wkbByteOrder {
  wkbXDR = 0,     // Big Endian
  wkbNDR = 1     // Little Endian
};
WKBPoint {
  byte     byteOrder;
  uint32   wkbType;     // 1=wkbPoint
  Point    point;
};
WKBLineString {
  byte     byteOrder;
  uint32   wkbType;     // 2=wkbLineString
  uint32   numPoints;
  Point    points[numPoints];
};

WKBPolygon    {
  byte                byteOrder;
  uint32            wkbType;     // 3=wkbPolygon
  uint32            numRings;
  LinearRing        rings[numRings];
};
WKBMultiPoint    {
  byte                byteOrder;
  uint32            wkbType;     // 4=wkbMultipoint
  uint32            num_wkbPoints;
  WKBPoint            WKBPoints[num_wkbPoints];
};
WKBMultiLineString    {
  byte              byteOrder;
  uint32            wkbType;     // 5=wkbMultiLineString
  uint32            num_wkbLineStrings;
  WKBLineString     WKBLineStrings[num_wkbLineStrings];
};

wkbMultiPolygon {
  byte              byteOrder;
```

```
  uint32            wkbType;     // 6=wkbMultiPolygon
  uint32            num_wkbPolygons;
  WKBPolygon        wkbPolygons[num_wkbPolygons];
};

WKBGeometry  {
  union {
    WKBPoint                point;
    WKBLineString           linestring;
    WKBPolygon              polygon;
    WKBMultiPoint           mpoint;
    WKBMultiLineString      mlinestring;
    WKBMultiPolygon         mpolygon;
  }
};
```

次の図は、NDR コーディングを使用した事前割り当てバイナリー表記の形状の例を示しています。

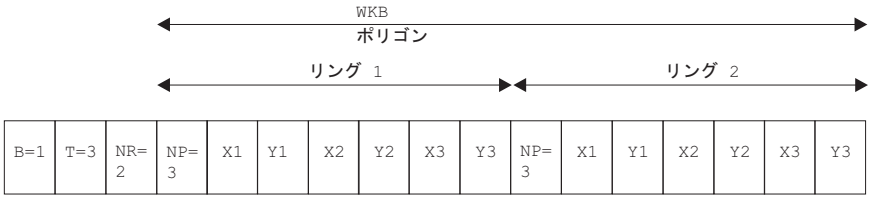


図 57. NDR フォーマットの形状表記： 2 つの線形を持つ (NR=2)、タイプがポリゴン (T=3) の (B=1)。ここで各リングは 3 つのポイントを持つ (NP=3)。

形状表記

形状表記は、ESRI により定義された、広範囲に使用されている業界標準です。形状表記の詳細な説明については、ESRI の Web サイト (http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf)を参照してください。

Geography Markup Language (GML) 表記

DB2 Spatial Extender には、Geography Markup Language (GML) 表記から形状を生成するいくつかの関数があります。

ジオグラフィー・マークアップ言語 (GML) は、OpenGIS コンソーシアムの「Geography Markup Language V2」仕様で定義された、ジオグラフィー情報の XML エンコード方式です。この OpenGIS コンソーシアムの仕様は、http://www.opengis.org/techno/implementation.htm から参照可能です。

第 26 章 サポートされる座標システム

このトピックでは、DB2 Spatial Extender によりサポートされる座標システムの構文を説明し、座標システムの値をリストしています。

座標システムの構文

空間参照系の事前割り当てテキスト表記は、座標システム情報の標準テキスト表記を提供します。事前割り当てテキスト表記は、OGC「Simple Features for SQL」仕様および ISO「SQL/MM Part 3: Spatial」標準に定義されています。

座標システムは、地理座標システム (緯度 - 経度) であるか、投影座標システム (X,Y) であるか、または地球の中心から見た (geocentric) 座標システム (X,Y,Z) です。座標システムは、複数のオブジェクトで構成されます。それぞれのオブジェクトは、大文字のキーワード (例えば、DATUM や UNIT など) を持ち、その後に続けて、オブジェクトの定義パラメーターをコンマで区切って大括弧内に指定します。オブジェクトのあるものは、他のオブジェクトから構成されるため、結果はネストされた構造になります。

注: 大括弧 [ ] を標準の括弧 ( ) で置き換えるのは自由であり、どちらのフォーマットの括弧も読めるはずです。

大括弧を使用した座標システムのストリング表記の EBNF (拡張バッカス正規形式) 定義は、次のとおりです (括弧の使用については、上の注を参照してください)。

```
<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>
```

座標システムのタイプは、使用されるキーワードで示されます。

PROJCS
データが投影座標にある場合は、データ・セットの座標システムは PROJCS キーワードで示します。

GEOGCS
データが地理座標のデータであれば、データ・セットの座標システムは GEOGCS キーワードで示します。

GEOCCS
データが地心から見た座標のデータであれば、データ・セットの座標システムは GEOCCS キーワードで示します。

PROJCS キーワードの後には、投影座標系を定義するすべての項目を指定します。どのオブジェクトであれ、最初の項目は必ず名前です。投影座標システムの名前の

© Copyright IBM Corp. 1998, 2009

後には、いくつかのオブジェクト (地理座標システム、マップ投影、1 つまたは複数のパラメーター、および線形測定単位) が続きます。投影座標システムはすべて、地理座標システムに基づいているので、このセクションは、投影座標システムに特有な項目を最初に説明します。例えば、NAD83 データ上の UTM ゾーン 10N は次のように定義されます。

```
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]]
```

代わって、名前および複数のオブジェクトが、地理座標システム・オブジェクト (データ、本初子午線、および角度測定単位) を定義します。

```
<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>
```

半長軸 (semi-major axis) はメートルで測定され、ゼロより大きくなければなりません。

NAD83 上の UTM ゾーン 10 の地理座標システム・ストリング:

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],
UNIT["Degree",0.0174532925199433]]
```

UNIT オブジェクトは角度測定単位または線形測定単位を表すことができます。

```
<angular unit> = <unit>
<linear unit> = <unit>
<unit> = UNIT["<name>", <conversion factor>]
<conversion factor> = <number>
```

変換係数は、単位あたりのメートル数 (線形単位の場合) またはラジアン数 (角度単位の場合) を指定し、ゼロより大きい値でなければなりません。

したがって、UTM ゾーン 10N の完全なストリング表記は次のようになります。

```
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]]
```

地心から見た座標システムは、地理座標システムに非常に似ています。

```
<geocentric cs> = GEOCCS["<name>", <datum>, <prime meridian>, <linear unit>]
```

サポートされる線形単位

表 59. サポートされる線形単位

単位	変換係数
Meter	1.0
Foot (International)	0.3048
U.S. Foot	12/39.37
Modified American Foot	12.0004584/39.37
Clarke's Foot	12/39.370432
Indian Foot	12/39.370141
Link	7.92/39.370432
Link (Benoit)	7.92/39.370113
Link (Sears)	7.92/39.370147
Chain (Benoit)	792/39.370113
Chain (Sears)	792/39.370147
Yard (Indian)	36/39.370141
Yard (Sears)	36/39.370147
Fathom	1.8288
Nautical Mile	1852.0

サポートされる角度単位

表 60. サポートされる角度単位

単位	緯度の有効範囲	経度の有効範囲	変換係数
ラジアン	–pi/2 ラジアン以上、pi/2 ラジアン以下	–pi ラジアン以上、pi ラジアン以下	1.0
度 (10進数)	–90 度以上、90 度以下	–180 度以上、180 度以下	pi/180
分 (10進数)	–5400 分以上、5400 分以下	–10800 分以上、10800 分以下	(pi/180)/60
秒 (10進数)	–324000 秒以上、324000 秒以下	–648000 秒以上、648000 秒以下	(pi/180)*3600
Gon	–100 グラジアン以上、100 グラジアン以下	–200 グラジアン以上、200 グラジアン以下	pi/200

表 60. サポートされる角度単位 (続き)

単位	緯度の有効範囲	経度の有効範囲	変換係数
Grad	–100 グラジアン以上、100 グラジアン以下	–200 グラジアン以上、200 グラジアン以下	pi/200

サポートされる回転楕円面

表 61. サポートされる回転楕円面

名前	半長軸	逆平坦化
Airy 1830	6377563.396	299.3249646
Airy Modified 1849	6377340.189	299.3249646
Average Terrestrial System 1977	6378135.0	298.257
Australian National Spheroid	6378160.0	298.25
Bessel 1841	6377397.155	299.1528128
Bessel Modified	6377492.018	299.1528128
Bessel Namibia	6377483.865	299.1528128
Clarke 1858	6378293.639	294.260676369
Clarke 1866	6378206.4	294.9786982
Clarke 1866 (Michigan)	6378450.047	294.978684677
Clarke 1880	6378249.138	293.466307656
Clarke 1880 (Arc)	6378249.145	293.466307656
Clarke 1880 (Benoit)	6378300.79	293.466234571
Clarke 1880 (IGN)	6378249.2	293.46602
Clarke 1880 (RGS)	6378249.145	293.465
Clarke 1880 (SGA 1922)	6378249.2	293.46598
Everest (1830 Definition)	6377299.36	300.8017
Everest 1830 Modified	6377304.063	300.8017
Everest Adjustment 1937	6377276.345	300.8017
Everest 1830 (1962 Definition)	6377301.243	300.8017255
Everest 1830 (1967 Definition)	6377298.556	300.8017
Everest 1830 (1975 Definition)	6377299.151	300.8017255
Everest 1969 Modified	6377295.664	300.8017

表 61. サポートされる回転楕円面 (続き)

名前	半長軸	逆平坦化
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
International 1924	6378388.0	297.0
International 1967	6378160.0	298.25
Krassowsky 1940	6378245.0	298.3
NWL 9D	6378145.0	298.25
NWL 10D	6378135.0	298.26
OSU 86F	6378136.2	298.25722
OSU 91A	6378136.3	298.25722
Plessis 1817	6376523.0	308.64
Sphere	6371000.0	0.0
Sphere (ArcInfo)	6370997.0	0.0
Struve 1860	6378298.3	294.73
Walbeck	6376896.0	302.78
War Office	6378300.0	296.0
WGS 1966	6378145.0	298.25
WGS 1972	6378135.0	298.26
WGS 1984	6378137.0	298.257223563

サポートされる測地データ

表 62. サポートされる測地データ

名前	測地系
Adindan	Lisbon
Afgooye	Loma Quintana
Agadez	Lome
Australian Geodetic Datum 1966	Luzon 1911
Australian Geodetic Datum 1984	Mahe 1971
Ain el Abd 1970	Makassar
Amersfoort	Malongo 1987
Aratu	Manoca
Arc 1950	Massawa
Arc 1960	Merchich
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967

表 62. サポートされる測地データ (続き)

名前	測地系
Bern 1938	Naparima 1972
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Bogota	Nord de Guerre
Bukit Rimpah	NGO 1948
Camacupa	Nord Sahara 1959
Campo Inchauspe	NSWC 9Z-2
Cape	Nouvelle Triangulation Francaise
Carthage	New Zealand Geodetic Datum 1949
Chua	OS (SN) 1980
Conakry 1905	OSGB 1936
Corrego Alegre	OSGB 1970 (SN)
Cote d'Ivoire	Padang 1884
Datum 73	Palestine 1923
Deir ez Zor	Pointe Noire
Deutsche Hauptdreiecksnetz	Provisional South American Datum 1956
Douala	Pulkovo 1942
European Datum 1950	Qatar
European Datum 1987	Qatar 1948
Egypt 1907	Qornoq
European Reference System 1989	RT38
Fahud	South American Datum 1969

表 62. サポートされる測地データ (続き)

名前	測地系
Gandajika 1970	Sapper Hill 1943
Garoua	Schwarzeck
Geocentric Datum of Australia 1994	Segora
Guyane Francaise	Serindung
Herat North	Stockholm 1938
Hito XVIII 1963	Sudan
Hu Tzu	Shan Tananarive 1925
Hungarian Datum 1972	Timbalai 1948
Indian 1954	TM65
Indian 1975	TM75
Indonesian Datum 1974	Tokyo
Jamaica 1875	Trinidad 1903
Jamaica 1969	Trucial Coast 1948
Kalianpur	Voirol 1875
Kandawala	Voirol Unifie 1960
Kertau	WGS 1972
Kuwait Oil Company	WGS 1972 Transit Broadcast Ephemeris
La Canoa	WGS 1984
Lake	Yacare
Leigon	Yoff
Liberia 1964	Zanderij

サポートされる本初子午線

表 63. サポートされる本初子午線

ロケーション	座標
Greenwich	0° 0’ 0″
Bern	7° 26’ 22.5″ E
Bogota	74° 4’ 51.3″ W
Brussels	4° 22’ 4.71″ E
Ferro	17° 40’ 0″ W

表 63. サポートされる本初子午線 (続き)

ロケーション	座標
Jakarta	106° 48’ 27.79″ E
Lisbon	9° 7’ 54.862″ W
Madrid	3° 41’ 16.58″ W
Paris	2° 20’ 14.025″ E
Rome	12° 27’ 8.4″ E
Stockholm	18° 3’ 29″ E

サポートされるマップ投影

表 64. シリンダー投影

シリンダー投影	疑似シリンダー投影
Behrmann	Craster parabolic
Cassini	Eckert I
Cylindrical equal area	Eckert II
Equirectangular	Eckert III
Gall’s stereographic	Eckert IV
Gauss-Kruger	Eckert V
Mercator	Eckert VI
Miller cylindrical	McBryde-Thomas flat polar quartic
Oblique	Mercator (Hotine) Mollweide
Plate-Carée	Robinson
Times	Sinusoidal (Sansom-Flamsteed)
Transverse Mercator	Winkel I

表 65. 円すい図法

名前	円すい図法
Albers conic equal-area	Chamberlin trimetric
Bipolar oblique conformal conic	Two-point equidistant
Bonne	Hammer-Aitoff equal-area
Equidistant conic	Van der Grinten I
Lambert conformal conic	Miscellaneous

表 65. 円すい図法 (続き)

名前	円すい図法
Polyconic	Alaska series E
Simple conic	Alaska Grid (Modified-Stereographic by Snyder)

表 66. マップ投影パラメーター

パラメーター	説明
central_meridian	x 座標の起点として選択した経度の線
scale_factor	Scale_factor は通常、マップ投影でのひずみの量を減らすために使用されます。
standard_parallel_1	全体的にひずみのない緯度の線。「真のスケールの緯度」にも使用される。
standard_parallel_2	全体的にひずみのない経度の線。
longitude_of_center	マップ投影の中心ポイントを定義する経度。
latitude_of_center	マップ投影の中心ポイントを定義する緯度。
longitude_of_origin	x 座標の起点として選択した経度。
latitude_of_origin	y 座標の起点として選択した緯度。
false_easting	すべての x 座標の値を正にするために、x 座標に追加する値。
false_northing	すべての y 座標を正にするために、y 座標に追加する値。
azimuth	斜めの投影の中心線を定義する、北東の角度
longitude_of_point_1	マップ投影に必要な最初のポイントの経度。
latitude_of_point_1	マップ投影に必要な最初のポイントの緯度。
longitude_of_point_2	マップ投影に必要な 2 番目のポイントの経度。
latitude_of_point_2	マップ投影に必要な 2 番目のポイントの緯度。
longitude_of_point_3	マップ投影に必要な 3 番目のポイントの経度。
latitude_of_point_3	マップ投影に必要な 3 番目のポイントの緯度。
landsat_number	Landsat サテライトの番号。
path_number	特定のサテライトの軌道パス番号。

表 66. マップ投影パラメーター (続き)

パラメーター	説明
perspective_point_height	マップ投影のパースペクティブ・ポイントの地上からの高さ。
fipszone	State Plane 座標システム・ゾーン番号。
zone	UTM ゾーン番号。

第 27 章 DB2 コントロール・センターからの空間タスク

Spatial Extender ユーザー・インターフェースで使用できるタスクは、タスクの単純化に役立ちます。

多くのタスクを、コマンド・プロンプトからコマンドとして、アプリケーションからストアード・プロシージャーとして、または DB2 コントロール・センターから実行できます。このセクションでは、DB2 コントロール・センターから実行できるタスクについて説明します。

座標システムの変更

座標システムを変更すると、形状の実際の位置が大きく変更され、それが役に立たなくなる場合があります。座標システムを変更する前に、その変更の効果を必ず理解してください。

「**名前**」以外の任意のフィールドを変更できます。

- **組織**と**組織 ID:** これらはオプション値です。これらのパラメーターは、両方とも NULL か、または両方とも NULL 以外である必要があります。パラメーターの組み合わせにより座標システムが一意的に識別されます。
- **定義 (Definition):** 定義は最大 2048 文字までです。座標システムを提供するベンダーは、通常この値を組み込んでいます。

座標システムの作成

通常は、既存の座標システムを使用します。

独自の座標システムを作成する必要がある場合は、指定する座標情報がすべて正確で、使用するジオブラウザーと互換性があることを確認してください。

- **名前:** 座標システムの識別に使用します。「組織」と「組織 ID」で座標システムを識別します。
- **組織**と**組織 ID:** これらはオプション値です。これらのパラメーターは、両方とも NULL か、または両方とも NULL 以外である必要があります。パラメーターの組み合わせにより座標システムが一意的に識別されます。
- **定義 (Definition):** 定義は最大 2048 文字までです。座標システムを提供するベンダーは、通常この値を組み込んでいます。

空間列の作成

既存の表に空間データを追加する場合は、空間列を作成し、その列を空間参照系に登録します。

「空間列の作成」ウィンドウを使用して空間列を作成するには、表から「空間列」ウィンドウを開く必要があります。

- **列 (Column):** 列の名前を入力します。

© Copyright IBM Corp. 1998, 2009

- **タイプ・スキーマとタイプ名:** リストから値を選択します。
- **空間参照系:** オプション: リストから空間参照系を選択します。列を測地参照系に登録するには、2000000000 から 2000001000 までの範囲の ID を持つ測地参照系を指定します。

空間インデックスの作成

空間グリッド・インデックスまたは測地ボロノイ・インデックスを空間列に作成できます。索引を作成し、DB2 によるデータの検索と取得を容易にすることでパフォーマンスを向上できます。

推奨: 索引が確実に正しい結果を戻すようにするため、索引を作成する空間列中のすべてのデータについて同じ座標システムを使用してください。空間列は、すべてのデータで必ず同じ空間参照系と、同じ座標システムが使用されるよう登録できます。

- **空間列:** 索引が作成される列を選択します。
- **最も細かいグリッド:** ボロノイ索引を作成するには、-1 を入力します。それ以外の場合は、0 より大きい数値を入力します。
- **中程度のグリッド:** ボロノイ索引を作成するには、-1 を入力します。中程度のグリッドを使用しないために 0 を入力するか、または最も細かいグリッドを超える数値を入力します。
- **最も粗いグリッド:** ボロノイ索引を作成するには、使用するセル構造の ID (1 から 14) を入力します。最も粗いグリッドを使用するには中程度のグリッドを使用する必要があります。この値は中程度のグリッドより大きくする必要があります。最も粗いグリッドを使用しないためには 0 を入力します。

ジオコーディングの実行

バッチ・モードで空間データをジオコーディングするには、DB2 コントロール・センターの「ジオコーディングの実行」ノートブックを使用します。

「基本」ページ

ジオコーダーを指定してから、結果列を選択します。「**結果タイプ**」フィールドは、指定するジオコーダーに基づいて自動的に入力されます。コミット有効範囲と WHERE 節を指定することで、バッチ・ジョブを詳細にカスタマイズできます。

「ジオコーダー・パラメーター (Geocoder Parameters)」ページ

独自の列名またはパラメーター値のいずれかを指定できます。

ジオコーディングのセットアップ

ジオコーディングのセットアップ時に、列をデータのジオコーディングに使用するジオコーダーと関連付け、ジオコーダーが後で使用できるように対応するジオコーディング・パラメーターおよびその他の関連情報をセットアップします。

「基本」ページ

セットアップするジオコーダーを選択します。「**結果タイプ**」フィールドは、指定するジオコーダーに基づいて自動的に入力されます。「**コミット有効範囲**」フィールドを使用して、コミットを実行する前にジオコーディングする行数を選択できます。

ヒント: リストからジオコーダーを選択して「**自動ジオコード**」を選択することで、空間列が更新されるときに自動的にジオコーディングされるようにできます。

「ジオコーダー・パラメーター (Geocoder Parameters)」ページ

独自の列名またはパラメーター値のいずれかを指定できます。

空間参照系の変更

注意: 説明されている属性以外の属性を変更すると、その空間参照系と関連付けられているすべての空間データの値が変更され、無効になる可能性があります。

- 「**ID**」フィールドに、変更する空間参照系の ID を入力します。測地参照系の場合、ID の値は 2000000318 から 2000001000 までの範囲でのみ変更できます。
- **オフセット (Offset):** オフセット・トランスフォーメーションの場合、X、Y、Z、および M 座標のスケール値とオフセット値を入力します。スケール値は測定の乗数値です。デフォルト値は 1 です。値を変更して、座標が小数値から整数に変換されるときの正確さを維持できます。オフセット値は、座標値および測定値の正の整数値を取得するためにそれぞれの値から減算される数値です。スケール値とオフセット値は、空間参照系を作成するために必要です。
- **エクステント (Extent):** エクステント・トランスフォーメーションの場合、X、Y、Z、および M 座標の最小値と最大値を入力します。スケール値は測定の乗数値です。デフォルト値は 1 です。値を変更して、座標が小数値から整数に変換されるときの正確さを維持できます。スケール値とエクステント値は、「**エクステント (Extent)**」ラジオ・ボタンを選択した場合にのみ必要です。

空間データのインポート

シェイプ・ファイルをインポートできます。

「基本」ページ

このページを使用して、ソース情報とターゲット情報を指定します。例外とメッセージ・ファイルを指定することもできます。これらはインポートが失敗した場合のトラブルシューティングで役立ちます。

詳細ページ

このページを使用して、表スペースの詳細、およびインポートがどのように進むかについての詳細を指定します。

付録 A. DB2 技術情報の概説

DB2 技術情報は、以下のツールと方法を介して利用できます。

- DB2 インフォメーション・センター
  - トピック (タスク、概念、およびリファレンス・トピック)
  - DB2 ツールのヘルプ
  - サンプル・プログラム
  - チュートリアル
- DB2 資料
  - PDF ファイル (ダウンロード可能)
  - PDF ファイル (DB2 PDF DVD に含まれる)
  - 印刷資料
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ

注: DB2 インフォメーション・センターのトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、ibm.com にある DB2 インフォメーション・センターを参照してください。

技術資料、ホワイト・ペーパー、IBM Redbooks® 資料などのその他の DB2 技術情報には、オンライン (ibm.com) でアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (http://www.ibm.com/software/data/sw-library/) にアクセスしてください。

資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、db2docs@ca.ibm.com まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

© Copyright IBM Corp. 1998, 2009

DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、DB2 ライブラリーについて説明しています。DB2 ライブラリーに関する詳細な説明については、www.ibm.com/shop/publications/order にある IBM Publications Center にアクセスしてください。英語および翻訳された DB2 バージョン 9.7 のマニュアル (PDF 形式) は、 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947 からダウンロードできます。

この表には印刷資料が入手可能かどうかが示されていますが、国または地域によっては入手できない場合があります。

資料番号は、資料が更新される度に大きくなります。資料を参照する際は、以下にリストされている最新版であることを確認してください。

注: *DB2 インフォメーション・センター*は、PDF やハードコピー資料よりも頻繁に更新されます。

表 67. DB2 の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
管理 API リファレンス	SC88-5883-00	入手可能	2009 年 8 月
管理ルーチンおよびビュー	SC88-5880-00	入手不可	2009 年 8 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻	SC88-5885-00	入手可能	2009 年 8 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻	SC88-5886-00	入手可能	2009 年 8 月
コマンド・リファレンス	SC88-5884-00	入手可能	2009 年 8 月
データ移動ユーティリティー ガイドおよびリファレンス	SC88-5903-00	入手可能	2009 年 8 月
データ・リカバリーと高可用性 ガイドおよびリファレンス	SC88-5904-00	入手可能	2009 年 8 月
データベース: 管理の概念および構成リファレンス	SC88-5870-00	入手可能	2009 年 8 月
データベースのモニタリング ガイドおよびリファレンス	SC88-5872-00	入手可能	2009 年 8 月
データベース・セキュリティー・ガイド	SC88-5905-00	入手可能	2009 年 8 月

表 67. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 Text Search ガイド	SC88-5902-00	入手可能	2009 年 8 月
ADO.NET および OLE DB アプリケーションの開発	SC88-5874-00	入手可能	2009 年 8 月
組み込み SQL アプリケーションの開発	SC88-5875-00	入手可能	2009 年 8 月
Java アプリケーションの開発	SC88-5878-00	入手可能	2009 年 8 月
Perl、PHP、Python および Ruby on Rails アプリケーションの開発	SC88-5879-00	入手不可	2009 年 8 月
SQL および外部ルーチンの開発	SC88-5876-00	入手可能	2009 年 8 月
データベース・アプリケーション開発の基礎	GI88-4201-00	入手可能	2009 年 8 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GI88-4202-00	入手可能	2009 年 8 月
グローバリゼーション・ガイド	SC88-5906-00	入手可能	2009 年 8 月
DB2 サーバー機能 インストール	GC88-5888-00	入手可能	2009 年 8 月
IBM データ・サーバー・クライアント機能 インストール	GC88-5889-00	入手不可	2009 年 8 月
メッセージ・リファレンス 第 1 巻	SC88-5897-00	入手不可	2009 年 8 月
メッセージ・リファレンス 第 2 巻	SC88-5898-00	入手不可	2009 年 8 月
Net Search Extender 管理およびユーザーズ・ガイド	SC88-5901-00	入手不可	2009 年 8 月
パーティションおよびクラスタリングのガイド	SC88-5907-00	入手可能	2009 年 8 月
pureXML ガイド	SC88-5895-00	入手可能	2009 年 8 月
Query Patroller 管理およびユーザーズ・ガイド	SC88-5908-00	入手不可	2009 年 8 月

表 67. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
Spatial Extender および Geodetic Data Management Feature ユーザーズ・ガイドおよびリファレンス	SC88-5900-00	入手不可	2009 年 8 月
SQL プロシージャー言語: アプリケーションのイネーブルメントおよびサポート	SC88-5877-00	入手可能	2009 年 8 月
SQL リファレンス 第 1 巻	SC88-5881-00	入手可能	2009 年 8 月
SQL リファレンス 第 2 巻	SC88-5882-00	入手可能	2009 年 8 月
問題判別およびデータベース・パフォーマンスのチューニング	SC88-5871-00	入手可能	2009 年 8 月
DB2 バージョン 9.7 へのアップグレード	SC88-5887-00	入手可能	2009 年 8 月
Visual Explain チュートリアル	SC88-5899-00	入手不可	2009 年 8 月
DB2 バージョン 9.7 の新機能	SC88-5893-00	入手可能	2009 年 8 月
ワークロード・マネージャー ガイドおよびリファレンス	SC88-5894-00	入手可能	2009 年 8 月
XQuery リファレンス	SC88-5896-00	入手不可	2009 年 8 月

表 68. DB2 Connect 固有の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 Connect Personal Edition インストールおよび構成	SC88-5891-00	入手可能	2009 年 8 月
DB2 Connect サーバー機能 インストールおよび構成	SC88-5892-00	入手可能	2009 年 8 月
DB2 Connect ユーザーズ・ガイド	SC88-5890-00	入手可能	2009 年 8 月

表 69. Information Integration の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
Information Integration: Administration Guide for Federated Systems	SC19-1020-02	入手可能	2009 年 8 月
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-04	入手可能	2009 年 8 月
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-02	入手不可	2009 年 8 月
Information Integration: SQL Replication Guide and Reference	SC19-1030-02	入手可能	2009 年 8 月
Information Integration: Introduction to Replication and Event Publishing	GC19-1028-02	入手可能	2009 年 8 月

DB2 の印刷資料の注文方法

DB2 の印刷資料が必要な場合、オンラインで購入することができますが、すべての国および地域で購入できるわけではありません。 DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。 *DB2 PDF ドキュメンテーション* DVD の一部のソフトコピー・ブックは、印刷資料では入手できないことに留意してください。例えば、「*DB2 メッセージ・リファレンス*」はどちらの巻も印刷資料としては入手できません。

DB2 PDF ドキュメンテーション DVD で利用できる DB2 の印刷資料の大半は、IBM に有償で注文することができます。国または地域によっては、資料を IBM Publications Center からオンラインで注文することもできます。お客様の国または地域でオンライン注文が利用できない場合、DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。 DB2 PDF ドキュメンテーション DVD に収録されている資料の中には、印刷資料として提供されていないものもあります。

注: 最新で完全な DB2 資料は、DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7) で参照することができます。

DB2 の印刷資料は以下の方法で注文することができます。

- 日本 IBM 発行のマニュアルはインターネット経由でご購入いただけます。詳しくは http://www.ibm.com/shop/publications/order をご覧ください。資料の注文情報にアクセスするには、お客様の国、地域、または言語を選択してください。その後、各ロケーションにおける注文についての指示に従ってください。
- DB2 の印刷資料を IBM 営業担当員に注文するには、以下のようにします。

1. 以下の Web サイトのいずれかから、営業担当員の連絡先情報を見つけてください。
   - IBM Directory of world wide contacts (www.ibm.com/planetwide)
   - IBM Publications Web サイト (http://www.ibm.com/shop/publications/order)。国、地域、または言語を選択し、お客様の所在地に該当する Publications ホーム・ページにアクセスしてください。このページから、「このサイトについて」のリンクにアクセスしてください。
2. 電話をご利用の場合は、DB2 資料の注文であることをご指定ください。
3. 担当者に、注文する資料のタイトルと資料番号をお伝えください。タイトルと資料番号は、510 ページの『DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)』でご確認いただけます。

コマンド行プロセッサーから SQL 状態ヘルプを表示する

DB2 製品は、SQL ステートメントの結果の原因になったと考えられる条件の SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

SQL 状態ヘルプを開始するには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate or ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。
例えば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

異なるバージョンの DB2 インフォメーション・センターへのアクセス

DB2 バージョン 9.7 のトピックを扱っている DB2 インフォメーション・センターの URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/です。

DB2 バージョン 9.5 のトピックを扱っている DB2 インフォメーション・センターの URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/です。

DB2 バージョン 9 のトピックを扱っている DB2 インフォメーション・センターの URL は http://publib.boulder.ibm.com/infocenter/db2luw/v9/です。

DB2 バージョン 8 のトピックについては、バージョン 8 のインフォメーション・センターの URL http://publib.boulder.ibm.com/infocenter/db2luw/v8/にアクセスしてください。

DB2 インフォメーション・センターでの希望する言語でのトピックの表示

DB2 インフォメーション・センターでは、ブラウザーの設定で指定した言語でのトピックの表示が試みられます。トピックがその指定言語に翻訳されていない場合は、 DB2 インフォメーション・センターでは英語でトピックが表示されます。

- Internet Explorer Web ブラウザーで、指定どおりの言語でトピックを表示するには、以下のようにします。
  1. Internet Explorer の「**ツール**」 -> 「**インターネット オプション**」 -> 「**言語...**」ボタンをクリックします。「言語の優先順位」ウィンドウがオープンします。
  2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
     - リストに新しい言語を追加するには、「**追加...**」ボタンをクリックします。

       注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。
     - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「**上に移動**」ボタンをクリックします。
  3. ブラウザー・キャッシュを消去してから、ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。
- Firefox または Mozilla Web ブラウザーの場合に、希望する言語でトピックを表示するには、以下のようにします。
  1. 「**ツール**」 -> 「**オプション**」 -> 「**詳細**」 ダイアログの「**言語**」セクションにあるボタンを選択します。「設定」ウィンドウに「言語」パネルが表示されます。
  2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
     - リストに新しい言語を追加するには、「**追加...**」ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
     - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「**上に移動**」ボタンをクリックします。
  3. ブラウザー・キャッシュを消去してから、ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。

ブラウザーとオペレーティング・システムの組み合わせによっては、オペレーティング・システムの地域の設定も希望のロケールと言語に変更しなければなりません。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新

ローカルにインストールされた DB2 インフォメーション・センターは、定期的に更新する必要があります。

始める前に

DB2 バージョン 9.7 インフォメーション・センターが既にインストールされている必要があります。詳しくは、「*DB2 サーバー機能 インストール*」の『DB2 セットアップ・ウィザードによる DB2 インフォメーション・センターのインストール』

のトピックを参照してください。インフォメーション・センターのインストールに適用されるすべての前提条件と制約事項は、インフォメーション・センターの更新にも適用されます。

このタスクについて

既存の DB2 インフォメーション・センターは、自動で更新することも。手動で更新することもできます。

- 自動更新 - 既存のインフォメーション・センターのフィーチャーと言語を更新します。自動更新を使用すると、更新中にインフォメーション・センターが使用できなくなる時間が最小限で済むというメリットもあります。さらに、自動更新は、定期的に実行する他のバッチ・ジョブの一部として実行されるように設定することができます。
- 手動更新 - 更新処理中にフィーチャーまたは言語を追加する場合に使用する必要があります。例えば、ローカルのインフォメーション・センターが最初は英語とフランス語でインストールされており、その後ドイツ語もインストールすることにした場合、手動更新でドイツ語をインストールし、同時に、既存のインフォメーション・センターのフィーチャーおよび言語を更新できます。しかし、手動更新ではインフォメーション・センターを手動で停止、更新、再始動する必要があります。更新処理の間はずっと、インフォメーション・センターは使用できなくなります。

手順

このトピックでは、自動更新のプロセスを詳しく説明しています。手動更新の手順については、『コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新』のトピックを参照してください。

コンピューターまたはイントラネット・サーバーにインストールされている DB2 インフォメーション・センターを自動で更新するには、次のようにします。

1. Linux オペレーティング・システムの場合、次のようにします。
   a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、/opt/ibm/db2ic/V9.7 ディレクトリーにインストールされています。
   b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。
   c. 次のように ic-update スクリプトを実行します。

      ```
      ic-update
      ```
2. Windows オペレーティング・システムの場合、次のようにします。
   a. コマンド・ウィンドウを開きます。
   b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、<Program Files>¥IBM¥DB2 Information Center¥Version 9.7 ディレクトリーにインストールされています (<Program Files> は「Program Files」ディレクトリーのロケーション)。

c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。

d. 次のように ic-update.bat ファイルを実行します。

```
ic-update.bat
```

結果

DB2 インフォメーション・センターが自動的に再始動します。更新が入手可能な場合、インフォメーション・センターに、更新された新しいトピックが表示されます。インフォメーション・センターの更新が入手可能でなかった場合、メッセージがログに追加されます。ログ・ファイルは、doc¥eclipse¥configuration ディレクトリーにあります。ログ・ファイル名はランダムに生成された名前です。例えば、1239053440785.log のようになります。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新

DB2 インフォメーション・センターをローカルにインストールしている場合は、IBM から資料の更新を入手してインストールすることができます。

ローカルにインストールされた DB2 インフォメーション・センターを手動で更新するには、以下のことを行う必要があります。

1. コンピューター上の DB2 インフォメーション・センターを停止し、インフォメーション・センターをスタンドアロン・モードで再始動します。インフォメーション・センターをスタンドアロン・モードで実行すると、ネットワーク上の他のユーザーがそのインフォメーション・センターにアクセスできなくなります。これで、更新を適用できるようになります。DB2 インフォメーション・センターのワークステーション・バージョンは、常にスタンドアロン・モードで実行されます。を参照してください。
2. 「更新」機能を使用することにより、どんな更新が利用できるかを確認します。インストールしなければならない更新がある場合は、「更新」機能を使用してそれを入手およびインストールできます。

   注: ご使用の環境において、インターネットに接続されていないマシンに DB2 インフォメーション・センターの更新をインストールする必要がある場合、インターネットに接続されていて DB2 インフォメーション・センターがインストールされているマシンを使用して、更新サイトをローカル・ファイル・システムにミラーリングしてください。ネットワーク上の多数のユーザーが資料の更新をインストールする場合にも、更新サイトをローカルにミラーリングして、更新サイト用のプロキシーを作成することにより、個々のユーザーが更新を実行するのに要する時間を短縮できます。
   更新パッケージが入手可能な場合、「更新」機能を使用してパッケージを入手します。ただし、「更新」機能は、スタンドアロン・モードでのみ使用できます。
3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の DB2 インフォメーション・センターを再開します。

注: Windows 2008、Windows Vista (およびそれ以上) では、このセクションの後の部分でリストされているコマンドは管理者として実行する必要があります。完全な

管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを開くには、ショートカットを右クリックしてから、 「**管理者として実行**」を選択します。

コンピューターまたはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターを更新するには、以下のようにします。

1. DB2 インフォメーション・センターを停止します。
   - Windows では、「**スタート**」 → 「**コントロール パネル**」 → 「**管理ツール**」 → 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**停止**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv97 stop
     ```
2. インフォメーション・センターをスタンドアロン・モードで開始します。
   - Windows の場合:
     a. コマンド・ウィンドウを開きます。
     b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、<Program Files>¥IBM¥DB2 Information Center¥Version 9.7 ディレクトリーにインストールされています (<Program Files> は「Program Files」ディレクトリーのロケーション)。
     c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。
     d. 次のように help_start.bat ファイルを実行します。

        ```
        help_start.bat
        ```
   - Linux の場合:
     a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、/opt/ibm/db2ic/V9.7 ディレクトリーにインストールされています。
     b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。
     c. 次のように help_start スクリプトを実行します。

        ```
        help_start
        ```

   システムのデフォルト Web ブラウザーが開き、スタンドアロンのインフォメーション・センターが表示されます。
3. 「**更新**」ボタン ( ) をクリックします。(ブラウザーで JavaScript™ が有効になっている必要があります。) インフォメーション・センターの右側のパネルで、「**更新の検索 (Find Updates)**」をクリックします。 既存の文書に対する更新のリストが表示されます。
4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「**更新のインストール**」をクリックします。
5. インストール・プロセスが完了したら、「**完了**」をクリックします。
6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。

- Windows の場合は、インストール・ディレクトリーの doc¥bin ディレクトリーにナビゲートしてから、次のように help_end.bat ファイルを実行します。

  ```
  help_end.bat
  ```

  注: help_end バッチ・ファイルには、help_start バッチ・ファイルを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。help_start.bat は、Ctrl-C や他の方法を使用して停止しないでください。
- Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help_end スクリプトを実行します。

  ```
  help_end
  ```

  注: help_end スクリプトには、help_start スクリプトを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。他の方法を使用して、help_start スクリプトを停止しないでください。

7. DB2 インフォメーション・センターを再開します。
   - Windows では、「**スタート**」 → 「**コントロール パネル**」 → 「**管理ツール**」 → 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**開始**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv97 start
     ```

更新された DB2 インフォメーション・センターに、更新された新しいトピックが表示されます。

DB2 チュートリアル

DB2 チュートリアルは、DB2 製品のさまざまな機能について学習するのを支援します。この演習をとおして段階的に学習することができます。

はじめに

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) から、このチュートリアルの XHTML 版を表示できます。

演習の中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、チュートリアルを参照してください。

DB2 チュートリアル

チュートリアルを表示するには、タイトルをクリックします。

「*pureXML ガイド*」の『**pureXML™**』
: XML データを保管し、ネイティブ XML データ・ストアに対して基本的な操作を実行できるように、DB2 データベースをセットアップします。

「*Visual Explain チュートリアル*」の『**Visual Explain**』
: Visual Explain を使用して、パフォーマンスを向上させるために SQL ステートメントを分析し、最適化し、調整します。

DB2 トラブルシューティング情報

DB2 データベース製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 ドキュメンテーション

トラブルシューティング情報は、「*DB2 問題判別ガイド*」、または*DB2 インフォメーション・センター*の『データベースの基本』セクションにあります。ここには、DB2 診断ツールおよびユーティリティーを使用して、問題を切り分けて識別する方法、最も頻繁に起こる幾つかの問題に対するソリューションについての情報、および DB2 データベース製品を使用する際に発生する可能性のある問題の解決方法についての他のアドバイスがあります。

DB2 Technical Support の Web サイト

現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、DB2 Technical Support の Web サイトを参照してください。Technical Support サイトには、最新の DB2 資料、TechNotes、プログラム診断依頼書 (APAR またはバグ修正)、フィックスパック、およびその他のリソースへのリンクが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

DB2 Technical Support の Web サイト (http://www.ibm.com/software/data/db2/support/db2_9/) にアクセスしてください。

ご利用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。 ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

付録 B. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。IBM 以外の製品に関する情報は、本書の最初の発行時点で入手可能な情報に基づいており、変更される場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

© Copyright IBM Corp. 1998, 2009

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを

経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。サンプル・プログラムは、現存するままの状態で提供されるものであり、いかなる種類の保証も提供されません。 IBM は、これらのサンプル・プログラムの使用から生ずるいかなる損害に対しても責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (*お客様の会社名*) (*西暦年*). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _*年を入れる*_. All rights reserved.

商標

IBM、IBM ロゴおよび ibm.com® は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml の「Copyright and trademark information」をご覧ください。

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標です。
- UNIX は、The Open Group の米国およびその他の国における登録商標です。
- Intel、Intel ロゴ、Intel Inside®、 Intel Inside ロゴ、Intel® Centrino®、Intel Centrino ロゴ、Celeron®、Intel® Xeon®、Intel SpeedStep®、Itanium®、Pentium® は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。
- Microsoft®、 Windows、 Windows NT®、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

© Copyright IBM Corp. 1998, 2009

［サ行］

[タ行]

[ハ行]

N

S

V

W

Printed in Japan

SC88-5900-00

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12

Spine information:

IBM DB2 9.7 for Linux, UNIX, and Windows Spatial Extender/Geodetic Data Management Feature ユーザーズ・ガイドおよびリファレンス