

IBM® DB2® Spatial Extender and Geodetic
Extender



ユーザース・ガイドおよびリファレンス

バージョン 8.2

IBM® DB2® Spatial Extender and Geodetic
Extender



ユーザース・ガイドおよびリファレンス

バージョン 8.2

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、『特記事項』に記載されている情報をお読みください。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。
<http://www.ibm.com/jp/manuals/main/mail.html>

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは
<http://www.ibm.com/jp/manuals/> の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

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： SC27-1226-01
IBM® DB2® Spatial Extender and Geodetic Extender
User's Guide and Reference
Version 8.2

発 行： 日本アイ・ピー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2004.8

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財)日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

© Copyright International Business Machines Corporation 1998, 2004. All rights reserved.

© Copyright IBM Japan 2004

目次

第 1 部 概要 1

第 1 章 DB2 Spatial Extender の概要 . . . 3

DB2 Spatial Extender の目的	3
空間データと測地データ	4
データによって地形を表す方法	4
空間データの性質	5
測地データの性質	6
空間データのソース	6
地形、空間情報、空間データおよび図形の組み合わせ方	8

第 2 章 図形について 11

図形	11
図形のプロパティ	13
タイプ	13
図形の座標	13
X 座標と Y 座標	14
Z 座標	14
M 座標	14
内部、境界、および外部	14
単純か非単純か	14
閉じた曲線	14
空か空でないか	14
最小外接長方形 (MBR)	15
ディメンション	15
空間参照系の ID	15

第 3 章 DB2 Spatial Extender の使用方法 17

DB2 Spatial Extender の使用法	17
DB2 Spatial Extender および関連機能に対するインターフェース	17
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	17

第 2 部 DB2 Spatial Extender のセットアップ 23

第 4 章 DB2 Spatial Extender 入門 . . . 25

Spatial Extender のセットアップとインストールステップ	25
Spatial Extender のセットアップとインストール	25
Spatial Extender をインストールするためのシステム要件	26
DB2 Spatial Extender for Windows のインストール	28
DB2 Spatial Extender for AIX のインストール	29
DB2 Spatial Extender for HP-UX のインストール	32

Solaris オペレーティング環境用 DB2 Spatial Extender のインストール	34
DB2 Spatial Extender for Linux のインストール	37
DB2 Spatial Extender のインストール環境の作成	39
Spatial Extender のインストールの検査	41
インストールの問題のトラブルシューティング	43
インストール後の考慮事項	43
ArcExplorer for DB2 のダウンロード	43
ジオコーダー参照データへのアクセス	44
DB2 Spatial Extender のデータと地図の入った CD	45

第 5 章 Spatial Extender 環境の DB2 Universal Database バージョン 8 へのマイグレーション 47

地理情報操作が可能になっているデータベースのマイグレーション	47
マイグレーション・メッセージ	48
db2se migrate_v82 コマンド	49

第 6 章 データベースのセットアップ . . . 51

空間データを収容するデータベースの構成	51
データベース構成パラメーターのチューニング	51
トランザクション・ログ特性のチューニング	51
アプリケーション・ヒープ・サイズのチューニング	53
アプリケーション制御ヒープ・サイズのチューニング	54

第 7 章 データベース用の地理情報リソースのセットアップ 55

データベースにリソースをセットアップする方法	55
データベースに提供されるリソースのインベントリ	55
データベースに対する地理情報操作の使用可能化	56
参照データでの作業	57
参照データ	57
参照データに対するアクセスのセットアップ	57
ジオコーダーの登録	58

第 3 部 空間データを使用するプロジェクトの作成 59

第 8 章 プロジェクト用の地理情報リソースのセットアップ 61

座標系の使用法	61
座標系	61
測地座標系	62
投影座標系	67
座標系を選択または作成する	69

目次

空間参照系のセットアップ方法	70	地理情報分析を行うための環境	127
空間参照系	70	空間処理関数による操作の例	127
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	71	照会を最適化するために索引を使用する関数	128
DB2 Spatial Extender とともに提供される空間参照系	73	第 13 章 DB2 Spatial Extender コマンド	131
座標データを整数にトランスフォームする変換因子	75	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	131
空間参照系の作成	77	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	141
スケール因数の計算	79	DB2 Spatial Extender 用のアプリケーションを作成する	141
最小および最大の座標と指標を判別する	80	DB2 Spatial Extender のヘッダー・ファイルを地理情報アプリケーションに組み込む	141
オフセット値の計算	81	アプリケーションから DB2 Spatial Extender のストアド・プロシージャを呼び出す	142
第 9 章 空間列のセットアップ	83	DB2 Spatial Extender サンプル・プログラム	143
空間列	83	第 15 章 DB2 Spatial Extender の問題の識別	151
表示可能な内容をもった空間列	83	DB2 Spatial Extender メッセージの解釈方法	151
空間データ	83	DB2 Spatial Extender ストアド・プロシージャ出力パラメーター	153
空間列を作成する	85	DB2 Spatial Extender 関数メッセージ	156
空間列の登録	87	DB2 Spatial Extender CLP メッセージ	157
第 10 章 空間列にデータを入れる	89	DB2 コントロール・センター・メッセージ	159
空間データのインポートおよびエクスポート方法	89	db2trc コマンドによる DB2 Spatial Extender の問題のトレース	160
空間データのインポートとエクスポートについて	89	管理通知ファイル	162
空間データのインポート	90	第 4 部 DB2 Geodetic Extender の使用	165
空間データのエクスポート	93	第 16 章 DB2 Geodetic Extender	167
ジオコーダーの使用法	95	DB2 Geodetic Extender	167
ジオコーダーとジオコーディング	95	DB2 Geodetic Extender を使用すべき場合と DB2 Spatial Extender を使用すべき場合	168
ジオコーディング操作のセットアップ	97	測地基準	168
自動的に実行されるジオコーダーのセットアップ	99	測地緯度と測地経度	169
バッチ・モードでのジオコーダーの実行	100	測地距離	171
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	103	測地領域	172
空間インデックスのタイプ	103	第 17 章 DB2 Geodetic Extender のセットアップ	175
空間グリッド・インデックス	104	DB2 Geodetic Extender のセットアップと使用可能化	175
空間グリッド・インデックスの生成	104	Informix Geodetic DataBlade から DB2 Geodetic Extender へのマイグレーション	176
照会中での空間処理関数の使用	105	空間列に測地データを入れる	183
照会による空間グリッド・インデックスの利用の方法	105	第 18 章 測地索引	185
索引レベル数と格子サイズに関する考慮事項	106	測地ポロノイ・インデックス	185
格子レベルの数	106	ポロノイ・セル構造	186
格子セル・サイズ	107	代替ポロノイ・セル構造を選択する際の考慮事項	187
地理情報格子索引の作成	111	測地ポロノイ・インデックスの作成	189
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	114		
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	115		
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	115		
空間グリッド・インデックス作成のための格子サイズの決定	116		
空間グリッド・インデックスに関する統計の分析	118		
gseidx コマンド	123		
空間列にアクセスするのにビューを使用する	126		
第 12 章 地理情報の分析および生成	127		

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	190
DB2 Geodetic Extender に提供されているボロノイ・セル構造	192
世界の人口密度を基準にしたセル構造 (ボロノイ ID: 1)	194
米国 (ボロノイ ID: 2)	195
カナダ (ボロノイ ID: 3)	196
インド (ボロノイ ID: 4)	197
日本 (ボロノイ ID: 5)	198
アフリカ (ボロノイ ID: 6)	199
オーストラリア (ボロノイ ID: 7)	200
ヨーロッパ (ボロノイ ID: 8)	201
北アメリカ (ボロノイ ID: 9)	202
南アメリカ (ボロノイ ID: 10)	203
地中海 (ボロノイ ID: 11)	204
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)	205
世界の工業国を対象 - G7 諸国 (ボロノイ ID: 13)	206
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)	207

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 209

x、y 属性の最小値、最大値	209
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	209
第 180 子午線をまたぐ直線セグメント	210
第 180 子午線をまたぐポリゴン	211
極点を囲むポリゴン	214
半球、赤道地帯、地球全体を表すポリゴン	215
DB2 Geodetic Extender によってサポートされる空間処理関数	218
DB2 Geodetic Extender のストアード・プロシージャとカタログ・ビュー	223
DB2 Geodetic Extender によってサポートされる測地系	223
測地回転楕円体	232

第 5 部 参照資料 235

第 20 章 ストアード・プロシージャ 237

GSE_export_sde	238
GSE_import_sde	240
ST_alter_coordsys	242
ST_alter_srs	244
ST_create_coordsys	248
ST_create_srs	250
ST_disable_autogeocoding	257
ST_disable_db	258
ST_drop_coordsys	260
ST_drop_srs	261
ST_enable_autogeocoding	262
ST_enable_db	264

ST_export_shape	266
ST_import_shape	270
ST_register_geocoder	278
ST_register_spatial_column	283
ST_remove_geocoding_setup	284
ST_run_geocoding	286
ST_setup_geocoding	289
ST_unregister_geocoder	293
ST_unregister_spatial_column	294

第 21 章 カタログ・ビュー 297

DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビュー	297
DB2GSE.ST_GEOMETRY_COLUMNS カタログ・ビュー	298
DB2GSE.ST_GEOCODER_PARAMETERS カタログ・ビュー	299
DB2GSE.ST_GEOCODERS カタログ・ビュー	301
DB2GSE.ST_GEOCODING カタログ・ビュー	301
DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビュー	303
DB2GSE.ST_SIZINGS カタログ・ビュー	304
DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビュー	305
DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビュー	307

第 22 章 空間処理関数: カテゴリおよび使用法 309

空間処理関数	309
図形値をデータ交換フォーマットに変換する空間処理関数	309
コンストラクター関数の概要	310
データ交換フォーマットに関して働く関数	310
座標から図形を作成する関数	311
例	312
事前割り当てテキスト (WKT) 表記への変換	313
事前割り当てバイナリー (WKB) 表記への変換	315
ESRI 形状表記への変換	316
Geography Markup Language (GML) 表記への変換	317
地勢を比較する関数	318
比較関数の概要	318
関数のリスト	320
ある図形が別の図形を含むかどうかをチェックする関数	320
ST_Contains	321
ST_Within	322
図形どうしが交差するかどうかをチェックする関数	323
ST_Intersects	324
ST_Crosses	324
ST_Overlaps	326
ST_Touches	327
図形のエンベロープを比較する関数	328
ST_EnvIntersects	328
ST_MBRIntersects	328
2 つのものが同一かどうかをチェックする関数	328

目次

ST_EqualCoordsys	328	ST_IsEmpty	337
ST_Equals	329	ST_IsSimple	337
ST_EqualSRS	329	図形の空間参照系を識別する関数	337
2つの図形が交差していないかをチェックする関数	330	ST_SrsId (ST_SRIDとも呼ばれる)	337
図形をDE-9IMパターン・マトリックス・ストリ		ST_SrsName	338
ングと比較する関数	330	既存の図形から新規図形を生成する関数	338
図形のプロパティについての情報を戻す関数	331	ある図形を別の図形に変換する関数	338
データ・タイプ情報を戻す関数	331	ST_Polygon	338
座標および指標に関する情報を戻す関数	331	ST_ToGeomColl	338
ST_CoordDim	332	ST_ToLineString	338
ST_IsMeasured	332	ST_ToMultiLine	339
ST_IsValid	332	ST_ToMultiPoint	339
ST_Is3D	332	ST_ToMultiPolygon	339
ST_M	332	ST_ToPoint	339
ST_MaxM	332	ST_ToPolygon	339
ST_MaxX	333	異なる空間構成を使用して新規図形を作成する関数	339
ST_MaxY	333	ST_Buffer	339
ST_MaxZ	333	ST_ConvexHull	340
ST_MinM	333	ST_Difference	341
ST_MinX	333	ST_Intersection	342
ST_MinY	333	ST_SymDifference	342
ST_MinZ	333	複数の図形から1つの図形を派生させる関数	343
ST_X	333	MBR 集約	343
ST_Y	333	ST_Union	343
ST_Z	333	和集約	343
図形内の図形に関する情報を戻す関数	333	指標に基づいて新規図形を派生させる関数	344
ST_Centroid	334	ST_FindMeasure (ST_LocateAlongとも呼ばれる)	344
ST_EndPoint	334	ST_MeasureBetween (ST_LocateBetweenとも呼ば	
ST_GeometryN	334	れる)	344
ST_LineStringN	334	既存の図形の修正フォームを作成する関数	345
ST_MidPoint	334	ST_AppendPoint	345
ST_NumGeometries	334	ST_ChangePoint	345
ST_NumLineStrings	334	ST_Generalize	345
ST_NumPoints	334	ST_M	345
ST_NumPolygons	335	ST_PerpPoints	345
ST_PointN	335	ST_RemovePoint	346
ST_PolygonN	335	ST_X	346
ST_StartPoint	335	ST_Y	346
境界、エンベロープ、およびリングに関する情報を		ST_Z	346
表示する関数	335	距離情報を戻す関数	346
ST_Boundary	335	索引情報を戻す関数	347
ST_Envelope	335	座標系間の変換	347
ST_EnvIntersects	336		
ST_ExteriorRing	336	第23章 空間処理関数: 構文およびパラ	
ST_InteriorRingN	336	メーター 349	
ST_MBR	336	空間処理関数: 考慮事項および関連データ・タイプ	349
ST_MBRIntersects	336	考慮する要素	349
ST_NumInteriorRing	336	ST_Geometryの値をサブタイプの値として扱う	350
ST_Perimeter	336	入力タイプ別の空間処理関数のリスト	351
図形のディメンションに関する情報を戻す関数	336	EnvelopesIntersect	354
ST_Area	336	MBR 集約	356
ST_Dimension	337	ST_AppendPoint	357
ST_Length	337	ST_Area	358
図形が閉じているか、空か、または単純であるかを		ST_AsBinary	362
示す関数	337	ST_AsGML	363
ST_IsClosed	337	ST_AsShape	364

ST_AsText	365	ST_MinM	447
ST_Boundary	366	ST_MinX	448
ST_Buffer	368	ST_MinY	450
ST_Centroid	371	ST_MinZ	451
ST_ChangePoint	372	ST_MLineFromText	453
ST_Contains	373	ST_MLineFromWKB	454
ST_ConvexHull	375	ST_MPointFromText	456
ST_CoordDim	376	ST_MPointFromWKB	457
ST_Crosses	377	ST_MPolyFromText	459
ST_Difference	379	ST_MPolyFromWKB	460
ST_Dimension	380	ST_MultiLineString	462
ST_Disjoint	381	ST_MultiPoint	464
ST_Distance	383	ST_MultiPolygon	465
ST_Edge_GC_USA	386	ST_NumGeometries	467
ST_Endpoint	390	ST_NumInteriorRing	468
ST_Envelope	391	ST_NumLineStrings	469
ST_EnvIntersects	393	ST_NumPoints	470
ST_EqualCoordsys	394	ST_NumPolygons	471
ST_Equals	395	ST_Overlaps	472
ST_EqualSRS	396	ST_Perimeter	474
ST_ExteriorRing	397	ST_PerpPoints	475
ST_FindMeasure または ST_LocateAlong	398	ST_Point	477
ST_Generalize	400	ST_PointFromText	480
ST_GeomCollection	402	ST_PointFromWKB	481
ST_GeomCollFromTxt	404	ST_PointN	482
ST_GeomCollFromWKB	405	ST_PointOnSurface	483
ST_Geometry	406	ST_PolyFromText	484
ST_GeometryN	408	ST_PolyFromWKB	485
ST_GeometryType	409	ST_Polygon	487
ST_GeomFromText	410	ST_PolygonN	489
ST_GeomFromWKB	411	ST_Relate	490
ST_GetIndexParms	412	ST_RemovePoint	491
ST_InteriorRingN	415	ST_SrsId、ST_SRID	493
ST_Intersection	416	ST_SrsName	494
ST_Intersects	418	ST_StartPoint	495
ST_Is3d	419	ST_SymDifference	496
ST_IsClosed	420	ST_ToGeomColl	498
ST_IsEmpty	422	ST_ToLineString	499
ST_IsMeasured	423	ST_ToMultiLine	500
ST_IsRing	424	ST_ToMultiPoint	501
ST_IsSimple	425	ST_ToMultiPolygon	502
ST_IsValid	426	ST_ToPoint	503
ST_Length	427	ST_ToPolygon	504
ST_LineFromText	429	ST_Touches	505
ST_LineFromWKB	430	ST_Transform	507
ST_LineString	431	ST_Union	508
ST_LineStringN	433	ST_Within	510
ST_M	434	ST_WKBToSQL	512
ST_MaxM	435	ST_WKTToSQL	513
ST_MaxX	437	ST_X	514
ST_MaxY	439	ST_Y	515
ST_MaxZ	440	ST_Z	517
ST_MBR	442	和集約	518
ST_MBRIntersects	443		
ST_MeasureBetween、ST_LocateBetween	444	第 24 章 トランスフォーム・グループ 521	
ST_MidPoint	446	トランスフォーム・グループ	521

目次

ST_WellKnownText トランスフォーム・グループ	521
ST_WellKnownBinary トランスフォーム・グループ	523
ST_Shape トランスフォーム・グループ	524
ST_GML トランスフォーム・グループ	525

第 25 章 サポートされるデータ・フォーマット 527

事前割り当てテキスト (WKT) 表記	527
事前割り当てバイナリー (WKB) 表記	532
形状表記	533
ジオグラフィー・マークアップ言語 (GML) 表記	534

第 26 章 サポートされる座標系 535

サポートされる座標系	535
座標系の構文	535
サポートされる角度単位	537
サポートされる回転楕円面	538
サポートされる測地データ	539
サポートされる本初子午線	541
サポートされるマップ投影	542

付録 A. 使用すべきでないストアード・プロシージャ 545

db2gse.gse_enable_autogc	545
db2gse.gse_enable_db	548
db2gse.gse_enable_idx	548
db2gse.gse_enable_sref	550
db2gse.gse_export_shape	551
db2gse.gse_disable_autogc	552
db2gse.gse_disable_db	554
db2gse.gse_disable_sref	555
db2gse.gse_import_shape	555
db2gse.gse_register_gc	557
db2gse.gse_register_layer	559
db2gse.gse_run_gc	564
db2gse.gse_unregist_gc	565
db2gse.gse_unregist_layer	566

付録 B. 使用すべきでないカタログ・ビュー 569

DB2GSE.COORD_REF_SYS	569
DB2GSE.GEOMETRY_COLUMNS	569
DB2GSE.SPATIAL_GEOCODER	570

DB2GSE.SPATIAL_REF_SYS	571
------------------------	-----

付録 C. 使用すべきでない空間処理関数 573

AsShape	574
GeometryFromShape	574
Is3d	574
IsMeasured	574
LineFromShape	575
LocateAlong	575
LocateBetween	575
M	576
MLine FromShape	576
MPointFromShape	576
MPolyFromShape	577
PointFromShape	577
PolyFromShape	577
ShapeToSQL	577
ST_GeomFromText	578
ST_GeomFromWKB	578
ST_LineFromText	578
ST_LineFromWKB	579
ST_MLineFromText	579
ST_MLineFromWKB	579
ST_MPointFromText	580
ST_MPointFromWKB	580
ST_MPolyFromText	580
ST_MPolyFromWKB	581
ST_OrderingEquals	581
ST_Point	581
ST_PointFromText	581
ST_PolyFromText	582
ST_PolyFromWKB	582
ST_Transform	582
ST_SymmetricDiff	583
Z	583

特記事項 585

商標	587
----	-----

索引 589

IBM と連絡をとる 597

製品情報	597
------	-----

第 1 部 概要

第 1 章 DB2 Spatial Extender の概要

この章では、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により、当製品の概要を紹介しています。

DB2 Spatial Extender の目的

地形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。地理的な特性 (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); たとえば、河川や森、山岳地帯など。
- スペース; たとえば、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; たとえば、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。たとえば、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 が、ユーザーに対して使用できるようにする情報を意味します。つまり、地形のロケーションに関する事実や数字のことです。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 (たとえば、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 (たとえば、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 (たとえば、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 (たとえば、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。たとえば、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ります。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。

DB2 Spatial Extender の概要

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。たとえば、図解するために、視覚化ツールは、DB2 Spatial Extender が生成する、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。

空間データと測地データ

以下に、空間情報を取得する際に生成し、格納し、操作するデータの概要について説明します。以下のトピックを取り上げます。

- データによって地形を表す方法
- 空間データと測地データの性質
- 空間データの作成方法

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を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。たとえば、4 ページの図 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。図 2 は、このような値を入れることを指定し、行を新たにした 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-Mullern	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

空間データは座標によって構成されます。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。

たとえば、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は、まとめて座標系といわれます。緯度と経度以外の値に基づく座標系もあります。こうした座標系には、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。詳細については、61 ページの『座標系』を参照してください。

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型です。

測地データの性質

DB2 Geodetic Extender は、DB2 データベースに測地データを保存するために、Spatial Extender と同じデータ・タイプと関数を使用します。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系 (61 ページの『座標系』を参照)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。地球を平面の地図で表現する Spatial Extender とは異なり、Geodetic Extender は、地球を、極や日付変更線などで途切れることのない連続した球体として扱うのです。平面の地図は、球面座標を平面座標に変換するために投影座標を必要とします。それに対し、Geodetic Extender は地球表面の楕円モデル上の緯度、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。

詳細については、168 ページの『DB2 Geodetic Extender を使用すべき場合と DB2 Spatial Extender を使用すべき場合』、209 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。

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 (4 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、5 ページの図 2 を「ジオコーディング実行前の」ピクチャー、7 ページの図 3 を「ジオコーディング実行後の」ピクチャーとしましょう。5 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。この結果が 7 ページの図 3 に示されています。

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、空間処理関数と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

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。これらの関数のうち、コンストラクターと呼ばれる関数は、入力として指定する値から空間データを生成することができます。その他の関数では、入力として既存の空間データが必要です。たとえば、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。詳細については、このセクションの最後に『関連事項』、『関連資料』として示した箇所を参照してください。

空間データのインポート

空間データを取得するための 3 つ目の方法は、外部データ・ソースによって提供されるファイルからインポートすることです。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。たとえば、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。たとえば、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、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。

関連概念:

- 167 ページの『DB2 Geodetic Extender』
- 168 ページの『DB2 Geodetic Extender を使用すべき場合と DB2 Spatial Extender を使用すべき場合』
- 309 ページの『空間処理関数』
- 61 ページの『座標系』
- 339 ページの『異なる空間構成を使用して新規図形を作成する関数』

関連資料:

- 209 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』
- 218 ページの『DB2 Geodetic Extender によってサポートされる空間処理関数』
- 368 ページの『ST_Buffer』
- 338 ページの『既存の図形から新規図形を生成する関数』

地形、空間情報、空間データおよび図形の組み合わせ方

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、地理的地形、空間情報、空間データ、および図形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区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地形（ここでは短く地形）と呼びます。

たとえば、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の結論は、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。たとえば、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知る必要があります。

空間情報テクノロジーの世界では、地形は図形 (*geometry*) と呼ばれるシンボルで表現されたものとするのが一般的です。図形は、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。このような図形をポリゴン (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。このような図形を折れ線 と呼びます。

図形は、それが表す地形についての事実に対応するプロパティを持ちます。これらのプロパティのほとんどは、数学的に表現できます。たとえば、地形の座標は、合わせて、その地形の対応する図形のプロパティの 1 つを構成します。もう 1 つのプロパティはディメンション 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、図形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。格納されているものは、単なるデータとしてではなく、実際の図形と見なします。住居地域には間口と奥行きがあるので、図形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図形として見られます。たとえば、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を持つ図形と見なされます。

第 2 章 図形について

この章では、図形と呼ばれる情報のエンティティを説明します。図形は座標からなり、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。

- 図形
- 図形のプロパティ

図形

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、この言葉には図形の意味もあり、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。「図形」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、図形の定義として使用される定義は「地勢のモデル」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。たとえば、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。たとえば、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否かの情報を戻すことができます。

ある図形が表す地形の座標は、その図形のプロパティと見なされます。図形の種類によっては、他のプロパティ、たとえば、区域、長さ、境界などを持つことがあります。

DB2 Spatial Extender がサポートする図形は、以下に示すような階層を形成します。この図形階層は、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。

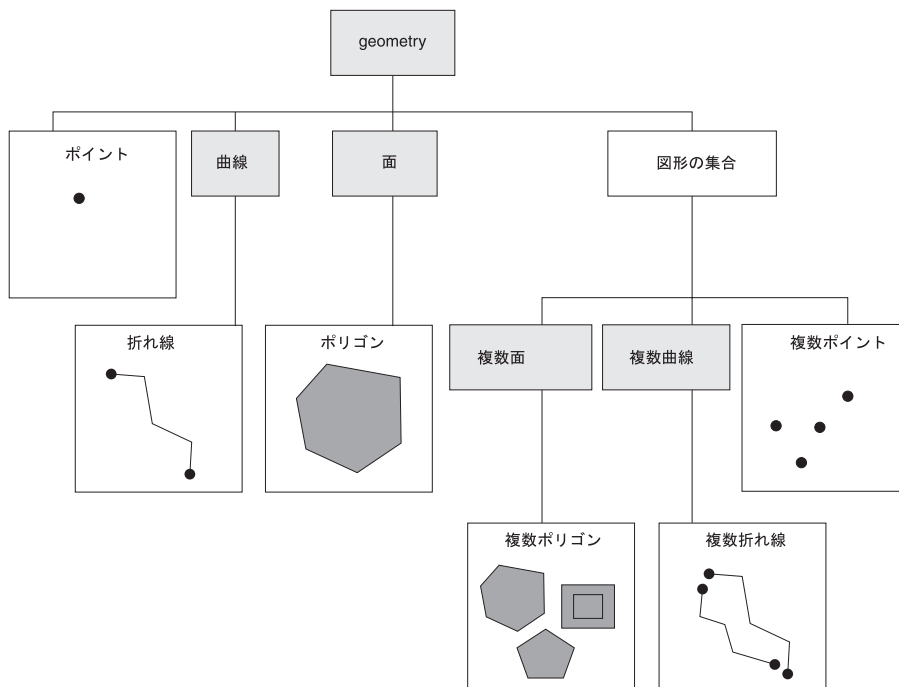


図 5. DB2 Spatial Extender のサポートする図形の階層： この図のインスタンス化可能図形には、図形がどのように可視的に描かれるかの例も含まれています。

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図形で示されているものです。

この図が示すように、図形 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、基本図形サブタイプと同種集合サブタイプの 2 つのカテゴリに分かれます。

基本図形には、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) と交差する交点を占めるものとして認識される個々の地形を表します。たとえば、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を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 (たとえば、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

複数の折れ線の集合です。複数折れ線は、複数の地形部分から構成される地形 (たとえば、水系や高速道路網) を表します。

複数ポリゴン

複数のポリゴン (面) 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 (たとえば、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、基本図形の集合です。同種集合は、基本図形のプロパティを共有するほかに、それぞれのプロパティもいくつか持っています。

関連概念:

- 4 ページの『空間データと測地データ』

図形のプロパティ

ここでは、図形のプロパティについて説明します。プロパティには、以下のものがあります。

- 図形が属しているタイプ
- 図形の座標
- 図形の内部、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
- 図形の最小外接長方形またはエンベロップ
- デイメンション
- 図形に関連付けられる空間参照系の ID

タイプ

各図形は、DB2 Spatial Extender によってサポートされる図形の階層中のタイプに属します。図形の階層の詳細については、11 ページの『図形』を参照してください。階層にある 7 つのタイプ (ポイント、折れ線、ポリゴン、図形の集合、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。

図形の座標

すべての図形には、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、空の図形には、座標はまったく含まれません。また、図形には、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。

図形について

- 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

図形の中には、高さまたは深度が関連付けられるものがあります。地形の図形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。たとえば、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

すべての図形は、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。図形の外部とは、その図形が占めないすべてのスペースです。図形の境界は、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、その図形が占めるスペース部分です。

単純か非単純か

ある種の図形サブタイプ (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。図形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、図形は単純であり、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、および空の図形は、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。マルチカーブは、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

図形がその中にポイントを 1 つも含んでいなければ、図形は空です。空の図形のエンベロップ、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。空の図形は、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)

図形の MBR は、最小と最大の (X,Y) 座標で形成される外接図形です。以下の特殊な場合を除いて、図形の 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

図形は、-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図形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、空間作業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 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 (CLP)。 *db2se CLP* と呼ばれます。
- DB2 Spatial Extender のストアード・プロシージャ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/>

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作業を簡単に説明します。説明には、作業を例示するシナリオが含まれています。作業は 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:

DB2 Spatial Extender の使用法

ここでは、DB2 Spatial Extender をセットアップするために行う作業をとりあげ、シナリオを使用して、架空の会社がそれぞれの作業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 Universal Database™ 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。たとえば、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 (たとえば、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 バージョン 8 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 7 をもっている場合は、空間データを DB2 のバージョン 8 にマイグレーションします。

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 8 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。米国以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 か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作業をとりあげます。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に設計されているので、これを使用します。

DB2 Spatial Extender の使用法

- 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 を使用しているため、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系の名前。
- 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時は、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一つである場合は、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すでに入っています。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 列に挿入されます。
 - 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 で空間データにすばや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。この作業には、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、たとえば、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。

関連タスク:

- 25 ページの『Spatial Extender のセットアップとインストール』

DB2 Spatial Extender の使用法

- 56 ページの『データベースに対する地理情報操作の使用可能化』
- 58 ページの『ジオコーダーの登録』
- 51 ページの『空間データを収容するデータベースの構成』
- 91 ページの『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』
- 92 ページの『SDE 転送データを新規の表または既存の表にインポートする』
- 97 ページの『ジオコーディング操作のセットアップ』
- 99 ページの『自動的に実行されるジオコーダーのセットアップ』
- 100 ページの『バッチ・モードでのジオコーダーの実行』
- 94 ページの『データを SDE 転送ファイルにエクスポートする』
- 69 ページの『座標系を選択または作成する』
- 87 ページの『空間列の登録』
- 85 ページの『空間列を作成する』
- 111 ページの『地理情報格子索引の作成』
- 142 ページの『アプリケーションから DB2 Spatial Extender のストアード・プロシージャを呼び出す』
- 141 ページの『DB2 Spatial Extender のヘッダー・ファイルを地理情報アプリケーションに組み込む』

関連資料:

- 131 ページの『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』

第 2 部 DB2 Spatial Extender のセットアップ

第 4 章 DB2 Spatial Extender 入門

この章では、Spatial Extender (AIX、HP-UX、Windows NT、Window 2000、Linux、Linux for z/OS 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 — ステップ

このセクションには、次のトピックの詳細が記述されています。

- Spatial Extender をインストールするためのシステム要件
- UNIX および Windows プラットフォームにインストールする方法
- DB2 Spatial Extender インスタンス環境を作成する方法
- Spatial Extender のインストールの検査
- インストール・サンプル・プログラムのトラブルシューティングのヒント

Spatial Extender のセットアップとインストール

DB2 Spatial Extender システムは、DB2 Universal Database、DB2 Spatial Extender から構成され、ほとんどのアプリケーションでは、地理情報ブラウザーも加わります。地理情報ブラウザーは必ずしも必要ではありませんが、地理情報照会の結果を(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。地理情報操作用に設定されたデータベースは、サーバー上に置かれます。クライアント・アプリケーションを使用し、DB2 Spatial Extender のストアード・プロシージャと地理情報照会を介して空間データにアクセスできます。さらに、スタンドアロン環境で DB2 Spatial Extender を構成することも可能です。この構成では、クライアントとサーバーの両方が同じマシンに存在することになります。クライアント/サーバーの構成でもスタンドアロンの構成でも、空間データは、ArcExplorer for DB2 や ESRI の ArcSDE と一緒に実行される ArcGIS ツール製品群などのような地理情報ブラウザーを使用して表示できます。

ArcExplorer for DB2 の無料コピーは、以下の IBM DB2 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/>

前提条件:

DB2 Spatial Extender をセットアップする前に、クライアントおよびサーバーで、DB2 ソフトウェアをインストールして構成しておく必要があります。

手順:

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。

2. Spatial Extender をインストールします。このステップは、オペレーティング・システムによって異なります。
 - Windows
 - AIX
 - HP-UX
 - Solaris オペレーティング環境
 - Linux
3. UNIX プラットフォーム:DB2 Spatial Extender のインスタンス環境を作成します。
4. インストールしたものを検証します。
5. 必要な場合は、トラブルシューティングのヒントを参考にして、問題を修正するために適切な処置をとります。
6. DB2 ドキュメントに自分のコンピューターからアクセスしたいが、まだ DB2 Information Center をインストールしていない場合は、Installing the DB2 Information Center (UNIX)」、あるいはInstalling the DB2 Information Center (Windows) を参照してください。DB2 Information Centerには、DB2 Universal Database と DB2 関連製品に関するドキュメントが収められています。

関連概念:

- 26 ページの『Spatial Extender をインストールするためのシステム要件』

関連タスク:

- 28 ページの『DB2 Spatial Extender for Windows のインストール』
- 29 ページの『DB2 Spatial Extender for AIX のインストール』
- 32 ページの『DB2 Spatial Extender for HP-UX のインストール』
- 34 ページの『Solaris オペレーティング環境用 DB2 Spatial Extender のインストール』
- 37 ページの『DB2 Spatial Extender for Linux のインストール』
- 39 ページの『DB2 Spatial Extender のインスタンス環境の作成』
- 41 ページの『Spatial Extender のインストールの検査』
- 43 ページの『インストールの問題のトラブルシューティング』
- 「Infrastructure Topics (DB2 Common Files)」の『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 「Infrastructure Topics (DB2 Common Files)」の『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』
- 43 ページの『ArcExplorer for DB2 のダウンロード』

関連資料:

- 45 ページの『DB2 Spatial Extender のデータと地図の入った CD』

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、32 ビット、64 ビット・バージョンの Windows[®]、AIX[®]、HP-UX、Solaris オペレーティング環境、Linux for Intel 上にインストールできます。Spatial Extender は、Linux for S/390[®] (32 ビット) をサポートしていますが、Linux for zSeries[®] (64 ビット) はサポートしていません。

ソフトウェア要件:

DB2 Spatial Extender をインストールするには、以下の DB2 ソフトウェアをサーバーにインストールし、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 Universal Database[™] Enterprise Server Edition Version 8.2 をシステムにインストールしておく必要があります。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 Universal Database 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。

地理情報クライアント・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地理情報クライアントが含まれています。DB2 Spatial Extender を AIX、HP-UX、Solaris オペレーティング環境、Linux for Intel、または Linux for S/390 上にインストールする場合は、DB2 サーバーをインストールするときに、管理またはアプリケーション開発クライアントとともに地理情報クライアントをインストールすることができます。これらのクライアントをインストールしない場合は、「カスタム」インストール・オプションを選択して手動で地理情報クライアントをインストールする必要があります。

ディスク・スペース要件:

Spatial Extender をインストールするには、システムは以下の表に示されたディスク・スペース要件を満たす必要があります。DB2 Spatial Extender のライブラリー・コードには、DB2 Geodetic Extender のためのコードも組み込まれていますが、Geodetic ライセンス・キーは組み込まれていません。

表 1. DB2 Spatial Extender のディスク・スペース要件

DB2 Spatial Extender ソフトウェア	ディスク・スペース
DB2 Spatial Extender のサーバー・ソフトウェア:	合計 596 MB のディスク・スペース:
<ul style="list-style-type: none"> Spatial Extender および Geodetic Extender サーバー・ライブラリー・コード、サンプル・ジオコーダー参照データ、および文書 	<ul style="list-style-type: none"> • 33 MB
<ul style="list-style-type: none"> オプションの、別の CD にある、ジオコーダー参照データ (米国) 	<ul style="list-style-type: none"> • 563 MB

表 1 は、DB2 Universal Database および DB2 Spatial Extender を、Windows に通常のインストールをする場合、または事前にコンポーネントを選択して AIX、

- 1 HP-UX、Solaris オペレーティング環境、Linux for Intel、および Linux for S/390 にインストールする場合に必要なディスク・スペースを示しています。異なるインストール・タイプを使用して、DB2 Spatial Extender をインストール、または DB2 Universal Database をインストールした場合は、ディスク・スペースの計算は異なるものになります。

システムがすべてのソフトウェアおよびディスク・スペース要件を満たす場合は、Spatial Extender をインストールすることができます。

DB2 Spatial Extender for Windows のインストール

この作業は、DB2 Spatial Extender をセットアップする作業の一部です。

「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、DB2 Spatial Extender を 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、インストール操作のヘルプが付いた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。また、このウィザードを使用すると、ユーザーやグループの自動作成、プロトコル構成、およびインスタンスの作成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 (Cancel)」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。

前提条件:

DB2 Spatial Extender 製品をインストールする前に、DB2 バージョン 8 のサーバー製品をインストールしておく必要があります。

手順:

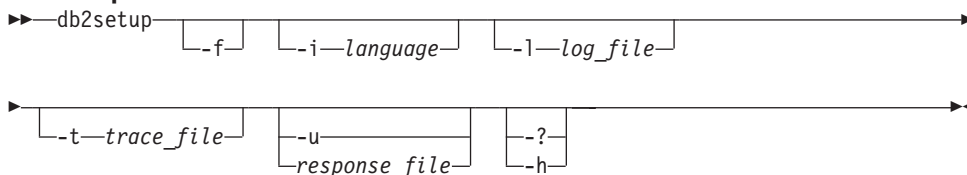
「DB2 セットアップ」ウィザードを使用して Spatial Extender for Windows をインストールするには、次のようにします。

- 1
1. Spatial Extender CD を CD ドライブに挿入します。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. 「製品のインストール (Install products)」をクリックします。
 3. インストールする製品として DB2 Spatial Extender を選択し、「次へ (NEXT)」をクリックします。「DB2 セットアップ」ウィザードが始まります。「次へ (NEXT)」をクリックします。セットアップを進める指示を出してくれる「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・ステップを進めます。インストールの間いつでも、「ヘルプ (Help)」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。

応答ファイルを使用して Spatial Extender for Windows をインストールするには、次のようにします。

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 を挿入します。詳しくは、「DB2 インストールおよび構成 補足」を参照してください。
3. コマンド・プロンプトから次のコマンドを出して、セットアップ・プログラムを実行します。

db2setup コマンド



コマンドの意味は次のとおりです。

-f インストール前にすべての DB2 プロセスを強制的に停止させます。

-i (language)
インストールを実行する際の言語の言語コードを示す 2 文字。

-l (log_file)
使用するログ・ファイルの絶対パスとファイル名。

-t (trace_file)
インストール・トレース情報の入る完全修飾ファイルを生成します。

-u (response_file)
完全修飾応答ファイル名を指定します。提供されているサンプル応答ファイルを変更または名前変更した場合は、このパラメーターが新しい名前に一致するようにします。このパラメーターは必須です。この応答ファイルは、DB2 Spatial Extender インストール CD の db2¥Windows¥samples¥db2gse.rsp に入っています。

-, -h 使用量についての情報を生成します。

4. インストールが終了したら、ログ・ファイルのメッセージをチェックします。

関連概念:

- 26 ページの『Spatial Extender をインストールするためのシステム要件』

関連タスク:

- 「インストールおよび構成 補足」の『応答ファイルの作成および編集 (Windows)』
- 41 ページの『Spatial Extender のインストールの検査』
- 43 ページの『インストールの問題のトラブルシューティング』

DB2 Spatial Extender for AIX のインストール

DB2 Spatial Extender for AIX をインストールするには、「DB2 セットアップ」ウィザード、db2_install スクリプト、またはシステム管理インターフェース・ツール (SMIT) を使用できます。

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、インストール操作のヘルプが付いた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。また、このウィザードを使用すると、ユーザーやグループの自動作成、プロトコル構成、およびインスタンスの作成を行うことができます。ウィザードを使用しない場合は、db2_install スクリプトまたは AIX のシステム管理インターフェース・ツール (SMIT) を使用して、Spatial Extender をインストールできます。SMIT を使用した Spatial Extender のインストールは、セットアップ・プロセスを変更する必要がある場合にのみ、経験豊かなユーザーに対してお勧めします。

前提条件:

AIX に Spatial Extender をインストールする前に、次のようにします。

- ご使用のシステムが、ソフトウェア、メモリー、ディスク・スペースのすべての要件を満たしていることを確認します。
- 構成パラメーターを更新し、AIX 上のすべての DB2 のクライアントとサーバーのためのシステムを再始動します。
- サーバー環境またはスタンドアロン環境でインストールする場合は、DB2 バージョン 8 のサーバー製品をインストールしておく必要があります。

注: DB2 Spatial Client がすでにインストールされているかどうかを調べてください。Spatial Extender のクライアントとサンプル・コンポーネントは、DB2 クライアント・サーバーで使用できます。これらの地理情報コンポーネントをインストールするには、DB2 カスタム・インストール・タイプを使用して、「クライアント・サポート (Client Support)」の下の「**Spatial Extender Client**」フィーチャーを選択し、「アプリケーション開発ツール (Application Development Tools)」の下の「**Spatial Extender サンプル (Spatial Extender Samples)**」フィーチャーを選択します。地理情報クライアントの機能のみが必要で、これらの地理情報コンポーネントを DB2 とともにインストール済みの場合は、DB2 Spatial Extender の以下のインストール手順を実行する必要はありません。

- root 権限をもっている必要があります。

手順:

「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には、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. Spatial Extender の CD を挿入し、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。CD のマウント方法については、「DB2 インストールおよび構成 補足」を参照してください。
3. インストールする製品として DB2 Spatial Extender を選択し、「次へ (NEXT)」をクリックします。
4. 「DB2 セットアップ」ウィザード・ウィンドウが開きます。セットアップを進める指示を出してくれる「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・ステップを進めます。インストールの間いつでも、「ヘルプ (Help)」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。

db2_install スクリプトを使用して DB2 Spatial Extender をインストールするには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. 適切な CD を挿入し、マウントします。
3. **/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 バージョン 8 CD のルート・ディレクトリーに入っています。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。
4. **DB2.GSE** と入力して、DB2 Spatial Extender をインストールします。

システム管理インターフェース・ツール (SMIT) を使用して DB2 Spatial Extender をインストールするには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. Spatial Extender の CD を挿入し、マウントします。
3. **smit install_latest** コマンドを入力します。
4. ソフトウェア・フィールドの「**INPUT デバイス/ディレクトリー (INPUT device/directory)**」に /cdrom/db2 を入力します。
5. 「**実行 (DO)**」をクリックするか Enter キーを押して、インストール・ディレクトリーが存在していることを検査します。
6. 「**インストールするソフトウェア (Software to install)**」フィールドで、クライアント・コンポーネントをインストールするのかサーバー・コンポーネントをインストールするのかを指定します。

注: DB2 Spatial Extender 用にインストールする必要がある全コンポーネントのリストは、DB2 Spatial Extender CD に入っている ComponentList.htm ファイルを参照してください。

7. 「**実行 (DO)**」をクリックするか、あるいは Enter キーを押します。インストール・パラメーターを確認するよう求められます。
8. 確認したら Enter キーを押します。
9. ログアウトします。

インストールが完了すると、Spatial Extender は、その他の DB2 製品と一緒に、/usr/opt/db2_08_01 ディレクトリーにインストールされます。

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。

関連概念:

- 26 ページの『Spatial Extender をインストールするためのシステム要件』

関連タスク:

- 「インストールおよび構成 補足」の『SMIT を使用して DB2 製品をインストールする (AIX)』

- 「DB2 Universal Database サーバー機能 概説およびインストール」の『CD-ROMのマウント (AIX)』
- 「インストールおよび構成 補足」の『db2_install スクリプトによる DB2 製品のインストール(UNIX)』
- 41 ページの『Spatial Extender のインストールの検査』
- 43 ページの『インストールの問題のトラブルシューティング』

DB2 Spatial Extender for HP-UX のインストール

Spatial Extender は、「DB2® セットアップ」ウィザード、db2_install スクリプトまたは **swinstall** コマンド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、インストール操作のヘルプが付いた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。また、このウィザードを使用すると、ユーザーやグループの自動作成、プロトコル構成、およびインスタンスの作成を行うことができます。ウィザードを使用しない場合は、db2_install スクリプトまたは **swinstall** コマンドを使用して、Spatial Extender for HP-UX をインストールできます。HP-UX **swinstall** コマンドを使用した Spatial Extender のインストールは、セットアップ・プロセスを変更する必要がある場合のみ、経験豊かなユーザーに対してお勧めします。

前提条件:

DB2 Spatial Extender for HP-UX をインストールする前に、次のことが必要です。

- ご使用のシステムが、ハードウェア、ソフトウェア、メモリーのすべての要件を満たしていることを確認します。
- DB2 バージョン 8 のサーバー製品をインストールしておく必要があります。

注: DB2 Spatial Client がすでにインストールされているかどうかを調べてください。Spatial Extender のクライアントとサンプル・コンポーネントは、DB2 クライアント・サーバーで使用できます。これらの地理情報コンポーネントをインストールするには、DB2 カスタム・インストール・タイプを使用して、「クライアント・サポート (Client Support)」の下の「**Spatial Extender Client**」フィーチャーを選択し、「アプリケーション開発ツール (Application Development Tools)」の下の「**Spatial Extender サンプル (Spatial Extender Samples)**」フィーチャーを選択します。地理情報クライアントの機能のみが必要で、これらの地理情報コンポーネントを DB2 とともにインストール済みの場合は、DB2 Spatial Extender の以下のインストール手順を実行する必要はありません。

- 構成パラメーターを更新し、HP-UX 上のすべての DB2 クライアントとサーバーのためのシステムを再始動します。
- root 権限をもっている必要があります。

手順:

「DB2 セットアップ」ウィザードを使用して Spatial Extender for HP-UX をインストールするには、次のようにします。

1. DB2 Spatial Extender の CD を挿入し、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 (NEXT)」をクリックします。「DB2 セットアップ」ウィザードが始まります。「次へ (NEXT)」をクリックします。セットアップを進める指示を出してくれる「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・ステップを進めます。インストールの間いつでも、「ヘルプ (Help)」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。

db2_install スクリプトを使用して Spatial Extender for HP-UX をインストールするには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. 適切な CD を挿入し、マウントします。
3. **./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 バージョン 8 CD のルート・ディレクトリーに入っています。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。
4. **DB2.GSE** と入力して、DB2 Spatial Extender をインストールします。

swinstall コマンドを使用して Spatial Extender for HP-UX をインストールするには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. Spatial Extender の CD を挿入し、マウントします。
3. 次のコマンドを使用して、swinstall プログラムを実行します。

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

このコマンドにより、「ソフトウェア選択 (Software Selection)」ウィンドウと「ソースの指定 (Specify Source)」ウィンドウがオープンします。必要な場合は、「ソースの指定 (Specify Source)」ウィンドウの「ソース・ホスト名 (Source host name)」フィールドで、値を変更します。

4. 「ソース保管パス (Source Depot Path)」フィールドで、*lcdrom/db2/hpux* と入力します。*lcdrom* は CD のマウント・ディレクトリーを表します。
5. 「OK」をクリックして、「ソフトウェア選択 (Software Selection)」ウィンドウに戻ります。「ソフトウェア選択 (Software Selection)」ウィンドウには、インストール可能なソフトウェアのリストがあります。
6. インストールのライセンス交付を受けた製品を選択します。
7. 「アクション (Actions)」メニューから「インストール用にマーク (Mark for Install)」を選択して、インストールする製品を選択します。次のようなメッセージが表示されます。

```
In addition to the software you just marked, other software was
automatically marked to resolve dependencies. This message will
not appear again.
```

8. 「OK」を選択します。

9. 「アクション (Actions)」メニューから「インストール (分析) (Install (analysis))」を選択して、製品のインストールを開始し、「インストール分析 (Install Analysis)」ウィンドウをオープンします。
10. 「状況 (Status)」フィールドに「準備完了 (Ready)」メッセージが表示されたら、「インストール分析 (Install Analysis)」ウィンドウで「OK」を選択します。
11. 「確認 (Confirmation)」ウィンドウで「はい (Yes)」を選択し、そのソフトウェアをインストールすることを確認します。

「状況 (Status)」フィールドに「準備完了 (Ready)」が表示され、「注釈 (Note)」ウィンドウがオープンするまでは、ソフトウェアがインストールされている間処理されているデータを表示する「インストール (Install)」ウィンドウに注目します。 `swinstall` プログラムはファイル・セットをロードし、ファイル・セット用のコントロール・スクリプトを実行します。

12. 「ファイル (File)」メニューから「終了 (Exit)」を選択して、`swinstall` から出ます。

インストールが完了すると、Spatial Extender は、その他の DB2 製品と一緒に、`/opt/IBM/db2/V8.1` ディレクトリーにインストールされます。

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。

関連概念:

- 26 ページの『Spatial Extender をインストールするためのシステム要件』

関連タスク:

- 「インストールおよび構成 補足」の『`swinstall` を使用して DB2 製品をインストールする (HP-UX)』
- 「DB2 Universal Database サーバー機能 概説およびインストール」の『HP-UX 上での CD-ROM のマウント』
- 「インストールおよび構成 補足」の『`db2_install` スクリプトによる DB2 製品のインストール(UNIX)』
- 41 ページの『Spatial Extender のインストールの検査』
- 43 ページの『インストールの問題のトラブルシューティング』

Solaris オペレーティング環境用 DB2 Spatial Extender のインストール

Spatial Extender は、「DB2® セットアップ」ウィザード、`db2_install` スクリプトまたは `pkgadd` コマンド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 DB2 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、インストール操作のヘルプが付いた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。また、このウィザードを使用すると、ユーザーやグループの自動作成、プロトコル構成、およびインスタンスの作成を行うことができます。ウィザードを使用しない場合は、`db2_install` スクリプトまたは Solaris オペレーティング環境 `pkgadd` コマン

ドを使用して、Spatial Extender をインストールできます。Solaris オペレーティング環境の **pkgadd** コマンドの使用は、セットアップ・プロセスで変更が必要である場合にのみ、経験豊かなユーザーに対してお勧めできます。

DB2 Spatial Extender は、Solaris オペレーティング環境ではパッケージと呼ばれる、さまざまな関数やコンポーネントから構成されています。 **pkgadd** コマンドを使用して Spatial Extender をインストールする場合は、必要なパッケージと使用するオプションの関数に関連するパッケージをインストールする必要があります。DB2 Spatial Extender 用にインストールする必要がある全パッケージのリストは、DB2 Spatial Extender CD に入っている ComponentList.htm ファイルに記載されています。ComponentList.htm ファイルは、/cdrom/db2/solaris に入っています。 /cdrom は DB2 Spatial Extender CD のマウント・ポイントです。

前提条件:

Solaris オペレーティング環境用の DB2 Spatial Extender をインストールする前に、次のことが必要です。

- ご使用のシステムが、ハードウェア、ソフトウェア、メモリーのすべての要件を満たしていることを確認します。
- サーバー環境またはスタンドアロン環境でインストールする場合は、DB2 バージョン 8 のサーバー製品をインストールしておく必要があります。

注: DB2 Spatial Client がすでにインストールされているかどうかを調べてください。Spatial Extender のクライアントとサンプル・コンポーネントは、DB2 クライアント・サーバーで使用できます。これらの地理情報コンポーネントをインストールするには、DB2 カスタム・インストール・タイプを使用して、「クライアント・サポート (Client Support)」の下の「**Spatial Extender Client**」フィーチャーを選択し、「アプリケーション開発ツール (Application Development Tools)」の下の「**Spatial Extender サンプル (Spatial Extender Samples)**」フィーチャーを選択します。地理情報クライアントの機能のみが必要で、これらの地理情報コンポーネントを DB2 とともにインストール済みの場合は、DB2 Spatial Extender の以下のインストール手順を実行する必要はありません。

- 構成パラメーターを更新し、Solaris オペレーティング環境のすべての DB2 のクライアントとサーバーのためのシステムを再始動します。
- root 権限をもっている必要があります。

手順:

「DB2 セットアップ」ウィザードを使用して DB2 Spatial Extender for Solaris オペレーティング環境をインストールするには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. DB2 Spatial Extender の CD を挿入し、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。CD のマウント方法については、「DB2 for UNIX 概説およびインストール」を参照してください。
3. インストールする製品として「**Spatial Extender**」を選択し、「次へ (NEXT)」をクリックします。

4. 「DB2 セットアップ」ウィザードが始まります。セットアップを進める指示を出してくれる「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・ステップを進めます。インストールの間いつでも、「ヘルプ (HELP)」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。

db2_install スクリプトを使用して DB2 Spatial Extender をインストールするには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. 適切な CD を挿入し、マウントします。
3. **./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 バージョン 8 CD のルート・ディレクトリーに入っています。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。
4. **DB2.GSE** と入力して、DB2 Spatial Extender をインストールします。

pkgadd コマンドを使用して DB2 Spatial Extender for Solaris をインストールするには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. DB2 Spatial Extender の CD を挿入し、マウントします。
3. インストールしようとする必須パッケージとオプション・パッケージを確認します。DB2 Spatial Extender 用にインストールする必要のある全コンポーネントのリストは、CD に入っている ComponentList.htm ファイルを参照してください。
4. 次のように入力して、インストールするパッケージごとに **pkgadd** コマンドを実行します。

```
pkgadd package_name
```

このコマンドの中で、*package_name* は、インストールしようとするパッケージです。

たとえば、Spatial Extender 基本サーバー・サポートをインストールする場合は、次のコマンドを入力して、db2gssg81 パッケージをインストールする必要があります。

```
pkgadd db2gssg81
```

インストールが完了すると、Spatial Extender ソフトウェアは、/opt/IBM/db2/V8.1 ディレクトリーにインストールされます。

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。

関連概念:

- 26 ページの『Spatial Extender をインストールするためのシステム要件』

関連タスク:

- 「インストールおよび構成 補足」の『pkgadd を使用して DB2 製品をインストールする (Solaris オペレーティング環境)』

- 「インストールおよび構成 補足」の『db2_install スクリプトによる DB2 製品のインストール(UNIX)』
- 「DB2 Universal Database サーバー機能 概説およびインストール」の『CD-ROM のマウント (Solaris)』
- 41 ページの『Spatial Extender のインストールの検査』
- 43 ページの『インストールの問題のトラブルシューティング』

DB2 Spatial Extender for Linux のインストール

DB2 Spatial Extender for Linux は、「DB2 セットアップ」ウィザード、db2_install スクリプト、または **rpm** コマンド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、インストール操作のヘルプが付いた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。また、このウィザードを使用すると、ユーザーやグループの自動作成、プロトコル構成、およびインスタンスの作成を行うことができます。ウィザードを使用しない場合は、db2_install スクリプトまたは **rpm** コマンドを使用して、Spatial Extender をインストールできます。Linux **rpm** コマンドを使用して Spatial Extender をインストールするのは、セットアップ・プロセスで手動制御部分を多くする必要がある場合に、経験豊かなユーザーのみに推奨できます。

前提条件:

DB2 Spatial Extender for Linux をインストールする前に、次のことが必要です。

- ご使用のシステムが、ハードウェア、ソフトウェア、メモリーのすべての要件を満たしていることを確認します。
- サーバー環境またはスタンドアロン環境にインストールする場合は、DB2 バージョン 8 サーバー製品がインストールされていることを確認します。

注:

- DB2 Spatial Client がすでにインストールされているかどうかを調べてください。DB2 Spatial Extender クライアントおよびサンプル・コンポーネントは、DB2 クライアントおよびサーバーで使用可能です。これらの地理情報コンポーネントをインストールするには、DB2 カスタム・インストール・タイプを使用して、「クライアント・サポート (Client Support)」の下の「Spatial Extender Client」フィーチャーを選択し、「アプリケーション開発ツール (Application Development Tools)」の下の「Spatial Extender サンプル (Spatial Extender Samples)」フィーチャーを選択します。地理情報クライアントの機能のみが必要で、これらの地理情報コンポーネントを DB2 とともにインストール済みの場合は、DB2 Spatial Extender の以下のインストール手順を実行する必要はありません。
- 構成パラメーターを更新し、Linux 上のすべての DB2 のクライアントとサーバーのためのシステムを再始動します。
- root 権限を持っていることを確認してください。

手順:

「DB2 セットアップ」ウィザードを使用して DB2 Spatial Extender をインストールするには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. DB2 Spatial Extender の CD を挿入し、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。CD のマウント方法については、「DB2 インストールおよび構成 補足」を参照してください。
3. 「製品のインストール (Install Products)」をクリックします。
4. インストールする製品として「Spatial Extender」を選択し、「次へ (NEXT)」をクリックします。
5. 「DB2 セットアップ」ウィザード・ウィンドウで、必要なオプションを選択します。インストールするオプションとしては、「DB2 Spatial Extender」または「DB2 Application Development Client」があります。
 - 地理情報クライアント機能のみ必要な場合は、「DB2 Application Development Client」を選択して、以下のフィーチャーを選択します。
 - 「クライアント・サポート (Client Support)」の下の「Spatial Extender Client」フィーチャー
 - オプション: 「アプリケーション開発ツール (Application Development Tools)」の下の「Spatial Extender サンプル (Spatial Extender Samples)」フィーチャー
 - 地理情報のサーバーとクライアントの両方の機能が必要な場合は、「DB2 Spatial Extender」を選択して、以下のフィーチャーを選択します。
 - 「サーバー・サポート (Server Support)」の下の「Spatial Extender Base Server サポート (Spatial Extender Base Server Support)」フィーチャー
 - 「Spatial Extender Client サポート (Spatial Extender Client Support)」の下の「Spatial Extender Client」フィーチャー
 - オプション: 「アプリケーション開発ツール (Application Development Tools)」の下の「Spatial Extender サンプル (Spatial Extender Samples)」フィーチャー

セットアップを進める指示を出してくれる「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・ステップを進めます。インストールの間いつでも、「ヘルプ (Help)」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。

db2_install スクリプトを使用して DB2 Spatial Extender をインストールするには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. 適切な CD を挿入し、マウントします。
3. `./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 バージョン 8 のルート・ディレクトリーに入っています。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。
4. **DB2.GSE** と入力して、DB2 Spatial Extender をインストールします。

rpm コマンドを使用して Spatial Extender for Linux をインストールするには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. DB2 for Linux のインストールのために、システムを使用可能にします。詳しくは、「DB2 インストールおよび構成 補足」を参照してください。
3. DB2 Spatial Extender の CD を挿入し、マウントします。
4. インストールする必須パッケージとオプション・パッケージを指定します。

注: DB2 Spatial Extender 用にインストールする必要がある全コンポーネントのリストは、DB2 Spatial Extender CD に入っている ComponentList.htm ファイルを参照してください。

5. インストールするパッケージごとに **rpm** コマンドを実行します。

```
rpm -ivh package_name
```

たとえば、サーバーをインストールする場合は、次のコマンドを入力して、IBM_db2gssg81-8.1.0-0.i386.rpm パッケージをインストールします。

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

インストールが完了すると、Spatial Extender は、その他の DB2 製品と一緒に、/opt/IBM/db2/V8.1 ディレクトリーにインストールされます。

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。

関連概念:

- 26 ページの『Spatial Extender をインストールするためのシステム要件』

関連タスク:

- 「インストールおよび構成 補足」の『rpm を使用して DB2 製品をインストールする (Linux)』
- 「DB2 Universal Database サーバー機能 概説およびインストール」の『CD-ROM のマウント (Linux)』
- 「インストールおよび構成 補足」の『db2_install スクリプトによる DB2 製品のインストール(UNIX)』
- 41 ページの『Spatial Extender のインストールの検査』
- 43 ページの『インストールの問題のトラブルシューティング』

DB2 Spatial Extender のインスタンス環境の作成

この作業は、Spatial Extender のセットアップ作業の一部です。

このセクションは、UNIX プラットフォームにしか適用できません。

DB2 Spatial Extender は、Spatial Extender のコードをインストールした後に作成されるあらゆる DB2 インスタンスと一緒に使用できます。

db2icrt コマンドは、新しい DB2 インスタンスを作成するために使用されます。DB2 Spatial Extender のインストール後に作成するすべての新しい DB2 インスタンスでは、インスタンス環境に DB2 Spatial Extender が含まれています。

Spatial Extender をインストールする前に作成した DB2 インスタンスでは、インスタンス環境に DB2 Spatial Extender が含まれていません。既存の DB2 インスタンスを更新するには、**db2iupdt** コマンドを使用します。DB2 Spatial Extender をインストールする前に DB2 コントロール・センターを使用して DB2 Administration server 用のインスタンスを作成した場合は、このインスタンスを更新する必要があります。

手順:

db2iupdt コマンドを使用してインスタンスを更新するには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. 次のコマンドを実行します。

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

コマンドの意味は次のとおりです。

DB2DIR

DB2 のインストール・ディレクトリー。

- AIX では、DB2 インストール・ディレクトリーは /usr/opt/db2_08_01 です。
- 他のすべての UNIX ベースのオペレーティング・システムでは、インストール・ディレクトリーは、/opt/IBM/db2/V8.1 です。

-a AuthType

インスタンスの認証タイプを表します。AuthType は、SERVER、CLIENT、DCS、SERVER_ENCRYPT、DCS_ENCRYPT のいずれかです。SERVER がデフォルトです。このパラメーターはオプションです。

-u FencedID

fenced ユーザー定義関数 (UDF) と fenced ストアード・プロシージャを実行するユーザー名を表します。DB2 クライアント上にインスタンスを作成する場合は、このフラグは必要ありません。作成した fenced ユーザーの名前を指定します。

InstName

インスタンスの名前を表します。インスタンス名は、インスタンス所有ユーザーの名前と同じでなければなりません。作成したインスタンス所有ユーザーの名前を指定します。インスタンスは、インスタンス所有ユーザーのホーム・ディレクトリーに作成されます。

db2icrt を使用してインスタンスを作成するには、次のようにします。

1. root 権限を持つユーザーとしてログインします。
2. 次のコマンドを実行します。

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

コマンドの意味は次のとおりです。

DB2DIR

DB2 のインストール・ディレクトリー。

- AIX では、DB2 インストール・ディレクトリーは /usr/opt/db2_08_01 です。

- 他のすべての UNIX ベースのオペレーティング・システムでは、インストール・ディレクトリーは、/opt/IBM/db2/V8.1 です。

-a AuthType

インスタンスの認証タイプを表します。AuthType は、SERVER、CLIENT、DCS、SERVER_ENCRYPT、DCS_ENCRYPT のいずれかです。SERVER がデフォルトです。このパラメーターはオプションです。

-u FencedID

fenced ユーザー定義関数 (UDF) と fenced ストアード・プロシージャを実行するユーザー名を表します。DB2 クライアント上にインスタンスを作成する場合は、このフラグは必要ありません。作成した fenced ユーザーの名前を指定します。

InstName

インスタンスの名前を表します。インスタンス名は、インスタンス所有ユーザーの名前と同じでなければなりません。作成したインスタンス所有ユーザーの名前を指定します。インスタンスは、インスタンス所有ユーザーのホーム・ディレクトリーに作成されます。

たとえば、サーバー認証を使用し、fenced ユーザーが db2fenc1、インスタンス所有ユーザーが db2inst1 の場合は、次のコマンドを使用して、AIX システム上にインスタンスを作成します。

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

インスタンスを作成した後は、ヘルス・モニターの通知を構成することもできます。この作業は、ヘルス・センターまたは CLP を使用して実行することができます。詳しくは、「DB2 インストールおよび構成 補足」を参照してください。

Spatial Extender のインストールの検査

この作業は、Spatial Extender のセットアップと構成作業の一部です。

DB2 Spatial Extender をインストールしたら、データベースを作成し、インストール検査プログラムを実行して、DB2 Spatial Extender が正常にインストールされて構成されていることを検証します。

インストールを検証するには、DB2 Spatial Extender のサンプル・プログラム runGseDemo を使用できます。runGseDemo プログラムは、インストールでの問題を明らかにするように設計されています。インストール検査中に、特定のシステムの問題の診断に役立つエラー・メッセージを受け取ることがあります。ほとんどのエラー・メッセージは、少数の一般的な問題が原因で出されます。こうしたエラーが発生しないようにするために、『前提条件』を参照してください。

このセクションでの検証ステップは、Windows、AIX、HP-UX、Solaris オペレーティング環境、Linux for Intel、および Linux for S/390 の各オペレーティング・システムで使用できます。

前提条件:

runGseDemo プログラムを実行する前に、以下のことを行ってください。

- DB2 Spatial Extender 製品を適切な環境にインストールしたことを確認する。

- 地理情報操作が関連付けられていない、新規のデータベースを使用する。
- UNIX (AIX、HP-UX、Solaris オペレーティング環境、Linux for Intel、および Linux for S/390) でのインストールでは、DB2 Spatial Extender のインスタンス環境が設定されていることを調べてください。インスタンスをチェックする **db2ilist** プログラムの実行方法については、「DB2 インストールおよび構成 補足」を参照してください。DB2 インスタンスを開始するために **db2start** コマンドを実行することが必要な場合があります。
- アプリケーション・ヒープ・サイズ用のデータベース構成パラメーター値を増やします。詳しくは、『インストールのトラブルシューティング』を参照してください。

手順:

インストールを検査するには、次のようにします。

1. UNIX の場合だけ: インスタンスの所有者としてログオンします。
2. データベースを作成します。

DB2 コマンド・ウィンドウをオープンして、以下を入力します。

```
db2 create database mydb
```

mydb はデータベース名です。

3. インストール検査プログラムを見つけます。
 - a. UNIX オペレーティング・システムの場合は、次のように入力します。

```
cd $HOME/sql1lib/samples/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。

- b. Windows の場合は、次のように入力します。

```
cd c:%Program Files%IBM%sql1lib%samples%spatial
```

c:%Program Files%IBM%sql1lib は、DB2 Spatial Extender をインストールしたディレクトリーです。

4. 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。たとえば、次のように入力します。

```
runGseDemo mydb userID password
```

mydb はデータベース名です。

検査プロセス中にエラー・メッセージを受け取る場合は、インストールのトラブルシューティングをする必要があります。

関連タスク:

- 43 ページの『インストールの問題のトラブルシューティング』
- 28 ページの『DB2 Spatial Extender for Windows のインストール』
- 29 ページの『DB2 Spatial Extender for AIX のインストール』
- 32 ページの『DB2 Spatial Extender for HP-UX のインストール』

- 34 ページの『Solaris オペレーティング環境用 DB2 Spatial Extender のインストール』
- 37 ページの『DB2 Spatial Extender for Linux のインストール』

インストールの問題のトラブルシューティング

Spatial Extender が適切にインストールされていることを検証するためにサンプル・プログラム runGseDemo を実行すると、以下のエラーになることがあります。

データベースがすでに地理情報処理可能になっている

インストールを検証するデータベースが新しいものであり、地理情報操作が関連付けられていないことをチェックしてください。これが該当する場合、サンプル・プログラムは失敗します。

サンプル・プログラムを実行するデータベースがすでに地理情報処理可能になっている場合は、次のエラー・メッセージを受け取ります。

```
Enabling database logtst...
Returning from ENABLE_DB:
Return code = -14
Return message text =
GSE0014E The database has already been enabled for spatial operations.
```

この問題を修正するには、データベースをドロップして、「Spatial Extender のインストールの検査」にあるステップを繰り返します。

アプリケーション・ヒープ・サイズ用のデータベース・マネージャー構成パラメーター値

APPLHEAPSZ が適切な値に設定されていない場合は、地理情報操作用にデータベースを使用可能にするようにしている間に、次のエラー・メッセージを受け取ります。

```
GSE0213N A bind operation failed.
SQLERROR = "SQL0001N Binding or precompilation
did not complete successfully.
SQLSTATE=00000".SQLSTATE=57011
```

アプリケーション・ヒープ・サイズ用のデータベース構成パラメーターの値を増やすには、次のように入力します。

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

2048 で十分でない場合は、APPLHEAPSZ パラメーターを 256 ずつ増やしてください。

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。

- ArcExplorer のダウンロード
- ジオコーダー参照データへのアクセス

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/>

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: DB2 Universal Database バージョン 8.1 は、IBM Software Developer's Kit (SDK) for Java バージョン 1.3.1. と共に出荷されています。 ArcExplorer for DB2 をインストールするには、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。CLASSPATH 環境変数を設定することを忘れないでください。

ジオコーダー参照データへのアクセス

DB2 Spatial Extender Geocoder Reference Data CD にあるジオコーダー参照データは、特に Spatial Extender が提供するジオコーダーで作業を行うために作成されています。米国の街路網についての基本地図のデータから構成されます。

DB2SE_USA ジオコーダーは、空間操作可能なデータベースにあるアドレスの座標を確定するために使用します。この基本地図データ全体をまとめて、参照データと呼びます。DB2SE_USA ジオコーダーは、データベース内のアドレス・データ (地理情報になっていない) を選んで、参照データと比較し、突き合わせます。そして、それを DB2 Spatial Extender に保管可能な座標に変換します。このプロセスを、ジオコーディング といいます。

CD で提供されている参照データには、EDGELocator.loc ファイルが含まれています。EDGELocator.loc ファイルは、特定の参照データを見つけるために、DB2SE_USA が使用します。たとえば、カリフォルニア、ケンタッキー、およびオレゴンのアドレスをジオコーディングする場合、DB2SE_USA ジオコーダーは、CD 上のロケーター・ファイルを使用して、アドレス・ロケーションを確定します。

手順:

ジオコーダー・データには CD から直接アクセスすることもできますし、データをハード・ディスクにコピーすることもできます。ジオコーダー・データ・ファイルを CD から DB2 Spatial Extender サーバー環境にコピーするには、この節で説明されているステップを実行してください。

UNIX オペレーティング・システムでは、次のようにします。

1. CD をマウントします。CD のマウント方法については、「*DB2 for UNIX 概説 およびインストール*」を参照してください。
2. ルート権限を持つユーザーとしてターゲット・サーバー・マシンにログインします。
3. 次のコマンドのいずれかを入力します。

- AIX の場合:

```
cp /cdrom/db2/* /usr/opt/db2_08_01/gse/refdata/
```

- その他のすべての UNIX プラットフォームの場合:

```
cp /cdrom/db2/* /opt/IBM/db2/V8.1/gse/refdata/
```

注: ジオコーダー・データ・ファイルは、ローカル・ドライブの任意のディレクトリーにコピーできます。指定するディレクトリーにそのファイルをコピーする場合は、新しいロケーションを指し示すように、ロケーター・ファイルを変更する必要があります。

4. ログアウトします。

Windows の場合は、「コマンド・ウィンドウ」と Windows エクスプローラのどちらでも使用できます。

「コマンド・ウィンドウ」を使用してジオコーダー・データにアクセスするには、次のようにします。

1. 「スタート」->「プログラム」->「IBM DB2」->「コマンド・ウィンドウ (Command Window)」の順にクリックします。
2. 次のコマンドを入力します。

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

d: は、ご使用の CD ドライブに該当する文字です。

注: ジオコーダー・データ・ファイルは、ローカル・ドライブの任意のディレクトリーにコピーできます。指定するディレクトリーにそのファイルをコピーする場合は、新しいロケーションを指し示すように、ロケーター・ファイルを変更する必要があります。

Windows エクスプローラを使用してジオコーダー・データにアクセスするには、次のようにします。

すべてのファイルを *d:%db2* から *c:%Program Files%IBM%sql%lib%gse%refdata* にコピーします。*d:* は CD ドライブであり、*c:%Program Files%IBM%sql%lib* は、DB2 がインストールされているディレクトリーです。

関連タスク:

- 25 ページの『Spatial Extender のセットアップとインストール』

DB2 Spatial Extender のデータと地図の入った CD

DB2 Spatial Extender は、データと地図の入った 7 枚の CD と一緒に出荷されます。

DB2 Spatial Extender Data and Maps 1 - 7 というラベルの付いたデータおよび地図は、7 枚の CD で提供されています。次の表は、各 CD にあるデータのサマリーを示しています。

表 2. データと地図の入った CD の説明

データと地図の入った CD	地図データ・タイプのサマリー
CD 1	ヨーロッパおよび世界
CD 2	カナダ、メキシコ、米国

表 2. データと地図の入った CD の説明 (続き)

データと地図の入った CD	地図データ・タイプのサマリー
CD 3	米国
CD 4	米国 (西部)
CD 5	米国 (中部)
CD 6	米国 (東部)
CD 7	米国 (南部)

ESRI で提供されるデータの詳細については、DB2 Spatial Extender Data and Maps CD にある ESRI ヘルプ・ファイルの `esridata.hlp` を参照してください。

- Windows の場合、`x: esridata.hlp` にあるヘルプ・ファイルを参照してください。`x:` は CD ドライブです。
- UNIX オペレーティング・システムの場合、CD の `/cdrom/esridata.hlp` にあるヘルプ・ファイルを参照するか、あるいは印刷してください。`/cdrom` はマウント・ポイントです。

関連タスク:

- 175 ページの『DB2 Geodetic Extenderのセットアップと使用可能化』

第 5 章 Spatial Extender 環境の DB2 Universal Database バージョン 8 へのマイグレーション

このセクションでは、DB2 Spatial Extender をバージョン 8.1 からバージョン 8.2 にマイグレーションする方法を説明しています。また、マイグレーション・ユーティリティを使用して、32 ビット環境から 64 ビット環境にマイグレーションする方法も説明しています。

地理情報操作が可能になっているデータベースのマイグレーション

DB2 Spatial Extender のバージョン 8.1 を使用している場合は、地理情報操作が可能になっているデータベースを DB2 Spatial Extender のバージョン 8.2 または DB2 Geodetic Extender のバージョン 8.2 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ます。ここでは、地理情報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 からマイグレーションする際に必要なステップを説明します。

前提条件:

マイグレーション処理を開始する前に以下を行ってください。

- マイグレーション・ユーティリティを実行する前に、データベースへのすべての接続を終了する。
- マイグレーション・プロセスを開始する前に、ご使用のシステムが、DB2 Spatial Extender Version 8.2 のインストール要件を満たしていることを確認する。
- データベースをバックアップするためには、データベースについての権限 SYSADM、SYSCTRL、または SYSMANT をもっている必要がある。
- データベースをマイグレーションするためには、SYSADM 権限をもっている必要がある。

手順:

DB2 Spatial Extender 環境をマイグレーションするには、次のようにします。

1. バージョン 8.1 のデータベースをバックアップします。データベースをバックアップする方法については、「DB2 インストールおよび構成 補足」を参照してください。
2. DB2 Universal Database のバージョン 8.2 と DB2 Spatial Extender のバージョン 8.2 をインストールします。
3. DB2 のインスタンスとデータベースをバージョン 8.1 からバージョン 8.2 にマイグレーションします。DB2 のインスタンスとデータベースをマイグレーションする方法の詳細については、「DB2 インストールおよび構成 補足」を参照してください。
4. Spatial Extender マイグレーション・ユーティリティを使用して、地理情報操作が可能なデータベースをバージョン 8.1 からバージョン 8.2 にマイグレーションします。

マイグレーション

- a. 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 migrate_v82** コマンドを使用してデータベースを移行します。

```
db2se migrate_v82 database_name userId user_id PW password
```

このコマンドの構文については、49 ページの『db2se migrate_v82 コマンド』を参照してください。

マイグレーション・メッセージ

マイグレーションに成功すると、次のメッセージが表示されます。

```
GSE0000I The operation was completed successfully
```

マイグレーションに失敗すると、次のメッセージが表示されます。

```
GSE9002N An error occurred during an attempt to perform  
Spatial Extender database migration.
```

注: マイグレーション中に、以下のエラーが発生する可能性があります。

- データベースは、現在、地理情報操作可能になっていません。
- データベースは、バージョン 8.1 の地理情報操作可能データベースではありません。
- データベースは、すでにバージョン 8.2 の地理情報操作可能データベースになっています。
- 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。
- 地理情報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。
- 移行できないユーザー・オブジェクトがあります。

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、および移行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。

関連タスク:

- 「*DB2 Universal Database* サーバー機能 概説およびインストール」の『DB2 の移行の前のデータベースのバックアップ』
- 「*DB2 Universal Database* サーバー機能 概説およびインストール」の『データベースの移行』
- 「*DB2 Universal Database* サーバー機能 概説およびインストール」の『インスタンスの移行 (UNIX)』
- 「*DB2 Universal Database* サーバー機能 概説およびインストール」の『DB2 UDB の移行 (Windows)』
- 「*DB2 Universal Database* サーバー機能 概説およびインストール」の『DB2 UDB サーバーの移行 (UNIX)』
- 「*DB2 Universal Database Personal Edition* 概説およびインストール」の『DB2 Personal Edition の移行 (Windows)』

- 「DB2 Universal Database Personal Edition 概説およびインストール」の『DB2 Personal Edition の移行 (Linux)』
- 「DB2 Universal Database Personal Edition 概説およびインストール」の『DB2 Personal Edition 上のデータベースの移行 (Windows)』
- 「DB2 Universal Database Personal Edition 概説およびインストール」の『DB2 Personal Edition 上のインスタンスおよびデータベースの移行 (Linux)』

関連資料:

- 49 ページの『db2se migrate_v82 コマンド』

db2se migrate_v82 コマンド

db2se migrate_v82 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 8.1 から 8.2 にマイグレーションします。

構文:

```

▶ db2se migrate_v82 database_name [-userId user_id -pw password]
                                     [-tableCreationParameters table_creation_parameters]
                                     [-force force_value] [-messagesFile messages_filename]

```

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。

-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。

-password

ユーザー・パスワード。

-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。

関連タスク:

- 175 ページの『DB2 Geodetic Extenderのセットアップと使用可能化』
- 25 ページの『Spatial Extender のセットアップとインストール』
- 56 ページの『データベースに対する地理情報操作の使用可能化』

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

DB2 Universal Database 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、地理情報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。地理情報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。地理情報操作のために、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があります。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。

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。

データベース構成パラメーターのチューニング

地理情報アプリケーションに影響を与えるデータベース構成パラメーターがいくつかあります。データベース構成パラメーターを変更するには、データベースに接続している必要があります。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。以下のセクションでは、地理情報アプリケーションに合うようにパラメーターをチューニングする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』
- 53 ページの『アプリケーション・ヒープ・サイズのチューニング』
- 54 ページの『アプリケーション制御ヒープ・サイズのチューニング』

トランザクション・ログ特性のチューニング

データベースを地理情報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地理情報操作に使用できるようにする
- ST_import_shape ストアド・プロシージャを使用して、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

データベースのセットアップ

現在、または将来、以上のどれかの使い方をする場合、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。この場合は、53 ページの『アプリケーション・ヒープ・サイズのチューニング』に進みます。

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。

表 3. 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
LOGFILSZ	ログ・ファイル・サイズを 4KB 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、データベースを地理情報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 Not enough log space is available to DB2.
```

手順:

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. GET DATABASE CONFIGURATION コマンドから、または DB2 コントロール・センターの「データベースの構成 (Configure Database)」ウィンドウからの出力を調べて、LOGFILSZ、LOGPRIMARY、LOGSECOND パラメーターの現行値を確認します。
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 LOGFILSZ 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。この場合は、53 ページの『アプリケーション・ヒープ・サイズのチューニング』に進みます。

4. LOGFILSZ 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。

- a. 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。
- b. データベースが明示的に活動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活動化されるか、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 (4KB 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少量です。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 (4KB ページの数) です。ST_enable_db ストアード・プロシージャを実行する場合は、この値は 2048 以上である必要があります。

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、データベースを地理情報操作に使用可能にしようとする、次のエラー・メッセージが出されます。

```
GSE0009N Not enough space is available in DB2's application heap.
```

```
GSE0213N A bind operation failed. SQLERROR = "SQL0001N Binding or precompilation did not complete successfully. SQLSTATE=00000".
```

手順:

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. GET DATABASE CONFIGURATION コマンドから、または DB2 コントロール・センターの「データベースの構成 (Configure Database)」ウィンドウからの出力を調べて、APPLHEAPSZ パラメーターの現行値を確認します。
2. 値を推奨値の 2048 か、それより大きな値に変更します。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活動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活動化されるか、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。

アプリケーション制御ヒープ・サイズのチューニング

すべての DB2 Spatial Extender アプリケーション、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、アプリケーション制御ヒープ・サイズの推奨値を使用すると、効率よく動作します。この特性は、APP_CTL_HEAP_SZ パラメーターで指定します。このパラメーターは、アプリケーション制御共有メモリの最大サイズを 4KB ページの数で指定します。アプリケーション制御ヒープは、この共有メモリから割り振られます。アプリケーションがアクティブになっているデータベース (あるいは、パーティション・データベース・システムの場合、アプリケーションがアクティブになっている各データベース・パーティション) で、各アプリケーションに対して 1 つのアプリケーション制御ヒープが割り振られます。ヒープが割り振られるのは、データベース (または、データベース・パーティション) のアプリケーションに対する要求を受け取った最初のエージェントが接続処理を行っているときです。ヒープは、同一アプリケーションのために働くエージェント間で情報を共有するために使用されます。(パーティション・データベース環境では、共有はデータベース・パーティション・レベルで行われます。データベース・パーティションを超えて共有が行われることはありません。) APP_CTL_HEAP_SZ パラメーターのデフォルト値は 128 です。

推奨: ほとんどの DB2 Spatial Extender アプリケーションで、1024 (4KB ページの数) 以上の APP_CTL_HEAP_SZ パラメーター値を使用してください。

APP_CTL_HEAP_SZ の値が適切でない場合、形状ファイルからデータベースにデータをインポートすると、次のエラー・メッセージが出されます。

```
GSE0214N An INSERT statement failed. SQLERROR = "SQL0973N Not enough storage
is available in the "APP_CTL_HEAP" heap to process the statement.
```

手順:

アプリケーション制御ヒープ・サイズを変更するには、次のようにします。

1. GET DATABASE CONFIGURATION コマンドから、または DB2 コントロール・センターの「データベースの構成 (Configure Database)」ウィンドウからの出力を調べて、APP_CTL_HEAP_SZ パラメーターの現行値を確認します。
2. 値を推奨値の 1024 (4KB ページの数) か、それより大きな値に変更します。次のコマンドを出します。db_name はデータベースを示します。

```
UPDATE DATABASE CONFIGURATION FOR db_name USING APP_CTL_HEAP_SZ 1024
```
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活動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活動化されるか、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 データベース用の地理情報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。

- 地理情報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 :
たとえば、データベースを管理するストアード・プロシージャ、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地理情報ユーティリティなど。
- 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作業を紹介します。その作業では、データベースを地理情報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作業は、データベースが地理情報操作をサポートできるようにすることです。地理情報操作とは、表に空間データを入れたり、地理情報の照会を処理するといった操作です。この作業では、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、作業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

データベースに対する地理情報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。たと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、地理情報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・プロシージャのどれか 1 つを呼び出します。
- 空間データ。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。たとえば、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。
- 地理情報グリッド索引。空間列にグリッド索引を定義できます。

データベース用の地理情報リソースのセットアップ

- 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。たとえば、図形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。
- 座標系の定義。
- 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、ストアド・プロシージャ、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地理情報操作の使用可能化

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作業のことを、通常、『データベースに対する地理情報操作の使用可能化』と呼びます。

前提条件:

データベースを地理情報操作に使用できるようにするには、あらかじめユーザー ID が、データベースに関して SYSADM 権限か DBADM 権限をもっている必要があります。

制約事項:

enable_db コマンドによって作成されたデータ・タイプのみを使用できます。

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地理情報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** コマンドを出す。
- db2gse.ST_enable_db ストアド・プロシージャを呼び出すアプリケーションを実行する。

注:

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。

関連概念:

- 55 ページの『データベースに提供されるリソースのインベントリ』

関連タスク:

- 141 ページの『DB2 Spatial Extender 用のアプリケーションを作成する』

関連資料:

- 131 ページの『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』
- 264 ページの『ST_enable_db』

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や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と指定された度合い、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。

参照データに対するアクセスのセットアップ

DB2SE_USA_GEOCODER の参照データは、Spatial Extender の入った CD のうちの 1 枚の中にあります。ここでは、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。

手順:

ジオコーダーのデフォルト参照データへのアクセスを準備するには、次のようになります。

1. 参照データを CD 上に置いておくか、ハード・ディスク上に保管するかどうかを決めます。CD 上に置いておく場合は、ハード・ディスクに保管した場合に占めるスペース (約 700 MB に相当) を節約できます。ハード・ディスク上に保管する場合は、CD から検索するよりも高速に検索することができるようになります。
2. 参照データをハード・ディスクに保管する場合は、次のようにします。
 - a. ハード・ディスクに、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。

データベース用の地理情報リソースのセットアップ

- b. データをハード・ディスクにコピーする。内容については、参照データに付いている README を参照してください。
 - c. コピーに成功したかどうかを確認する。UNIX 上でデータが適切にロードされたか検査するには、`$DB2INSTANCE/sqlib/gse/refdata/` ディレクトリーの中を探してください。Windows NT 上でデータが適切にロードされたか検査するには、`%DB2PATH%\gse\refdata\` ディレクトリーの中を探してください。
3. 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

データベースを地理情報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。

前提条件:

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 SYSADM 権限か、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 (Register Geocoder)」ウィンドウから登録する。
- `db2se register_gc` コマンドを出す。
- `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。

関連概念:

- 95 ページの『ジオコーダーとジオコーディング』

第 3 部 空間データを使用するプロジェクトの作成

第 8 章 プロジェクト用の地理情報リソースのセットアップ

データベースを地理情報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系、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。

- 座標系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。

座標系の使用法

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系の 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系に、要件を満たすものがない場合は、要件を満たす座標系を作成します。ここでは、座標系の概念を説明し、座標系を選択して使用する方法、および新しい座標系を作成する方法を紹介します。

座標系

座標系は、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。たとえば、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。

測地座標系

地理座標系 は、地球上の位置を確定するのに 3 ディメンションの球体の表面を利用する参照系です (70 ページの『空間参照系』を参照)。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。

投影座標系

投影座標系 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。この座標系は、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、投影変換によって地理座標に関連付けられます。

関連概念:

- 62 ページの『測地座標系』
- 67 ページの『投影座標系』

関連資料:

- 535 ページの『サポートされる座標系』

測地座標系

地理座標系は、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 (70 ページの『空間参照系』を参照)。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。その点の持つ値の単位としては、次のようなものが使用できます。

- 地理座標系が DB2[®] Geodetic Extender の認識する SRID (Spatial Reference System Identifier) を持っている場合には、線形単位。
- 座標系が DB2 Geodetic Extender の認識しない SRID を持っている場合には、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

それぞれの単位の値の範囲については、535 ページの『サポートされる座標系』を参照してください。

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

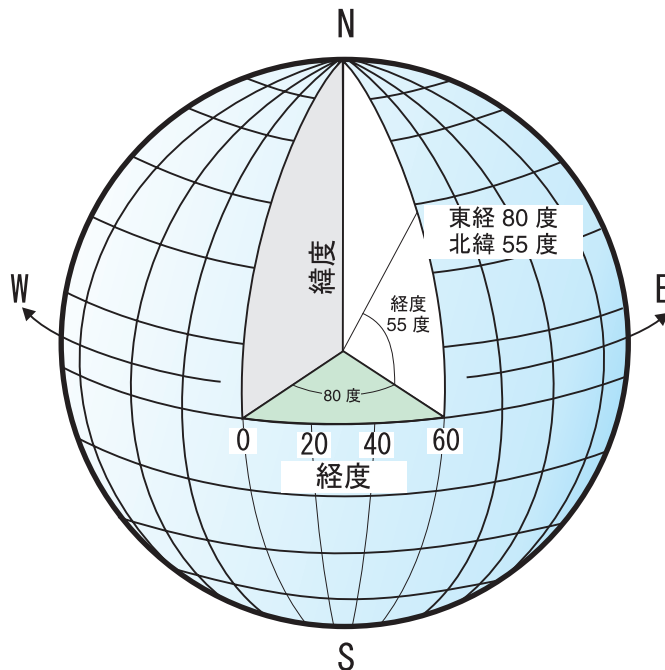


図 6. 地理座標系

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。各極が

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持ち、赤道より南の地点は、0 度から -90 度までの負の緯度を持ちます。

図 7 は緯線を表しています。

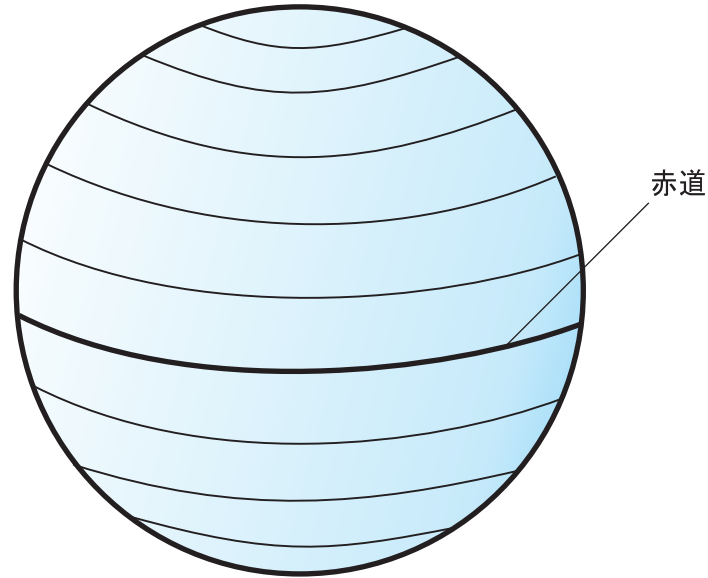


図 7. 緯線

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、ポゴタ、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持ちます。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持ちます。

64 ページの図 8 は経線を表しています。

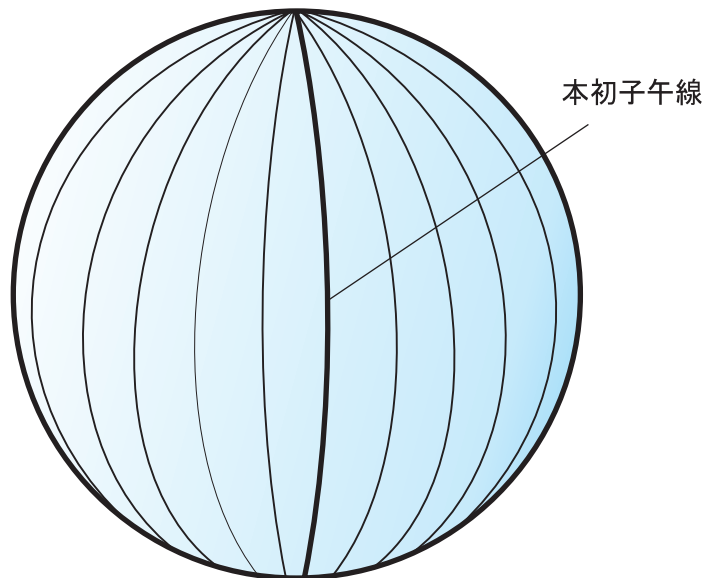


図8. 経線

経線と緯線は、地球を取り囲む経緯網と呼ばれる格子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。

65 ページの図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。

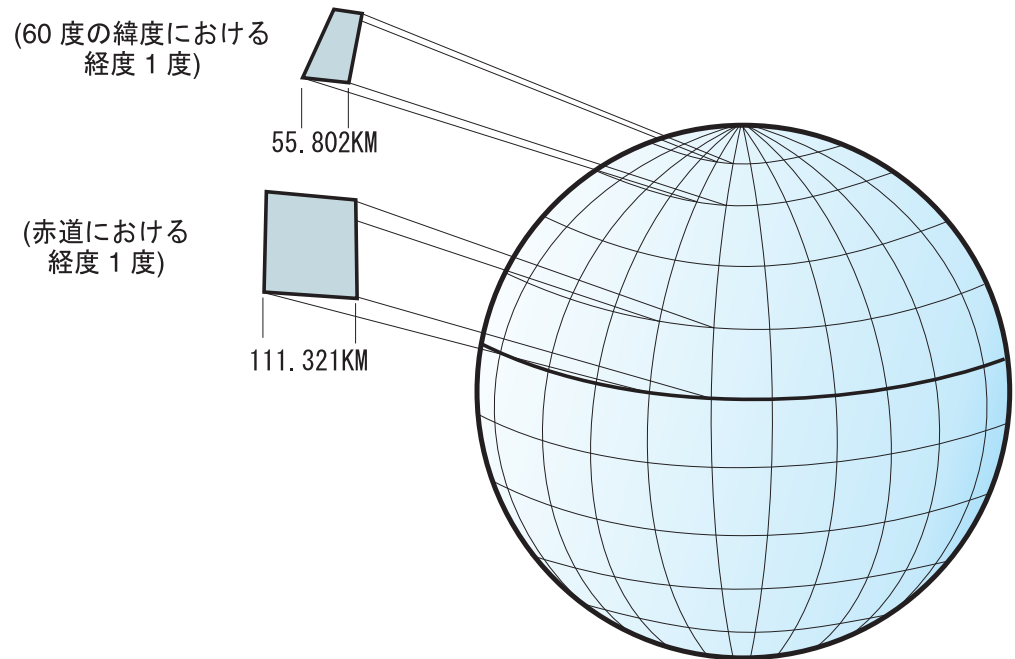
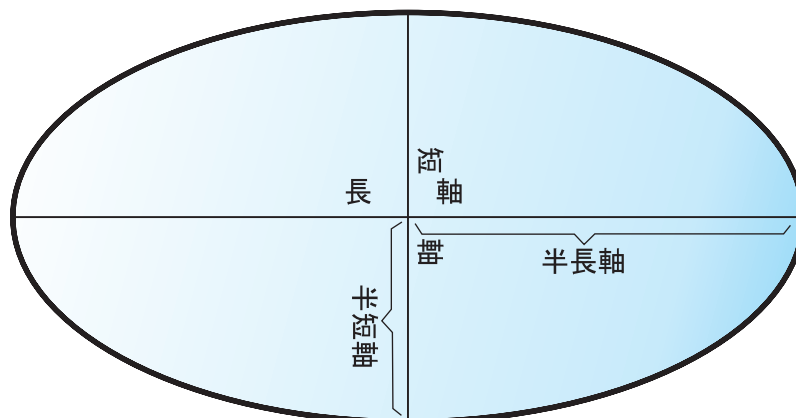
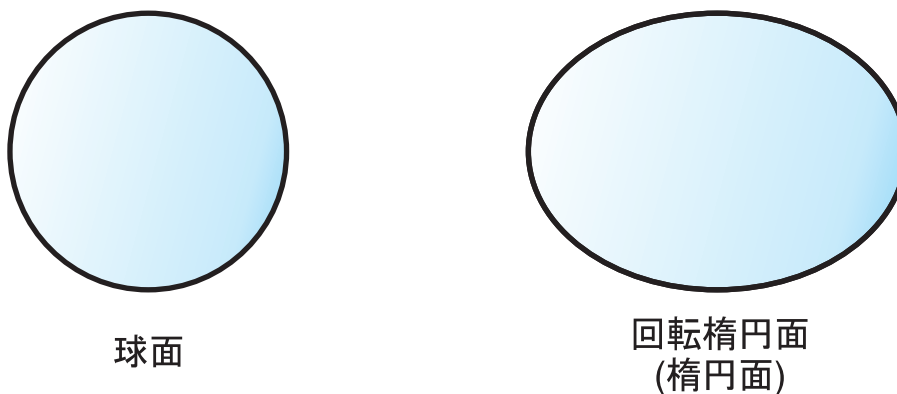


図9. 経緯網上のロケーションによる長さの違い

座標系は、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次元図形です。

66ページの図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。



楕円の長軸と短軸

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、座標系の測定は、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。楕円面の詳細については、535 ページの『サポートされる座標系』を参照してください。

67 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。

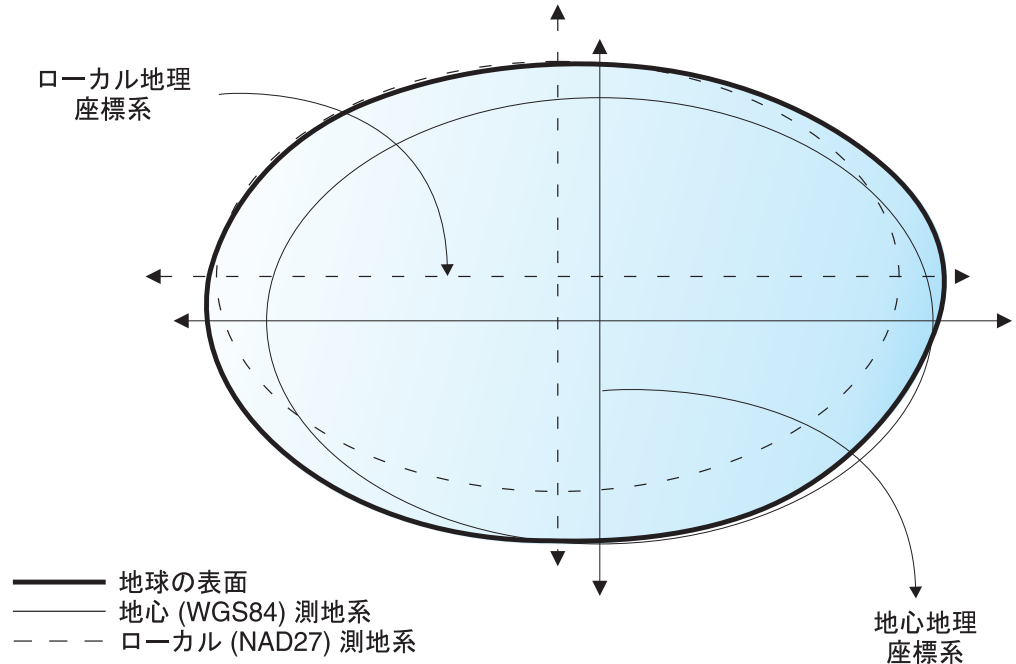


図 11. 測地系の配置

測地系を変更すると、必ず、地理座標系が変更され、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。

投影座標系

投影座標系は、地球を、平面的な 2 ディメンションで表現したものです。投影座標系は、球面や回転楕円面の地理座標系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系を 2 ディメンションの平面の投影座標系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の組み合わせは、扱わ

プロジェクト用の地理情報リソースのセットアップ

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、距離をゆがめません。正積図法の例として、*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、*Mercator* と *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、*Sinusoidal* 図法と *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、*Lambert Equal Area Azimuthal* 図法と *Azimuthal Equidistant* 図法があります。

関連概念:

- 62 ページの『測地座標系』
- 61 ページの『座標系』

関連タスク:

- 141 ページの『DB2 Spatial Extender 用のアプリケーションを作成する』

関連資料:

- 131 ページの『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』
- 248 ページの『ST_create_coordsys』
- 535 ページの『サポートされる座標系』

座標系を選択または作成する

データベースを地理情報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。プロジェクトを計画するには、まず、どの座標系を使用するかを決めます。以下のオプションがあります。

- DB2 Spatial Extender に付いてくる座標系を使用するか、ユーザーが作成した座標系を使用できます。DB2 Spatial Extender では、2000 を超える座標系を使用できます。以下のものがあります。
 - DB2 Spatial Extender で、『Unspecified』と呼ばれる座標系。以下の場合に、この座標系を使用します。
 - 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。たとえば、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、この座標系を使用します。たとえば、以下のような場合です。
 - DB2 Spatial Extender に付いてくる『Maps and Data』CD から、米国の空間データをインポートする場合。
 - 米国内のアドレスをジオコーディングするために、DB2 Spatial Extender に付いてくるジオコーダーを使用する計画がある場合。

これらの座標系の詳細を参照したり、DB2 Spatial Extender に付いてくるその他の座標系の内容や他のユーザーが作成した座標系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。

- 座標系を作成できます。

前提条件:

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

手順:

以下のどれかの方法によって、座標系を作成できます。

- DB2 コントロール・センターの「座標系の作成 (Create Coordinate System)」ウィンドウから作成する。
- db2se コマンド行プロセッサから、**db2se create_cs** コマンドを出す。
- db2se.ST_create_coordsys ストアード・プロシージャを呼び出すアプリケーションを実行する。

関連概念:

- 61 ページの『座標系』

関連タスク:

- 141 ページの『DB2 Spatial Extender 用のアプリケーションを作成する』

関連資料:

- 131 ページの『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』
- 248 ページの『ST_create_coordsys』

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作業および、新しく作成する作業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系の名前。
- 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 まででなければなりません。DB2® Geodetic Extender は、318 の定義済み測地参照系 (SRS) を提供しています。詳細については、167 ページの『DB2 Geodetic Extender』を参照してください。

地理情報列に保管される座標を包含するスペースの定義:

地理情報列の座標は、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の広がるスペースは、**地理情報エクステン**トと呼ばれます。たとえば、座標が地理情報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。洪水地帯の地理情報エクステン

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。地理情報列に 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。

地理情報エクステンツという用語は、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、地理情報エクステンツ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地理情報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、はらん平面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、座標系のほとんどの座標は小数值であり、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、スケール因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が得られます。(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、スケール因数は、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。

関連概念:

- 75 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因子』
- 61 ページの『座標系』

関連タスク:

- 71 ページの『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』
- 77 ページの『空間参照系の作成』

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

使用する座標系を決定すると、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 Extender 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。

プロジェクト用の地理情報リソースのセットアップ

手順:

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、以下の点を考慮してください。

1. 以下の質問の答えを考える。

- 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。

デフォルトの空間参照系のベースになる座標系としてどのようなものがあるかは、73 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。

- 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、223 ページの『DB2 Geodetic Extender によってサポートされる測地系』を参照してください。

- デフォルトの空間参照系に関連付けられている変換の因数は、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因数を使用して、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因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. 75 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因子』の内容を確認する。
 - b. これらの因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 より大きくなる場合は、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、77 ページの『空間参照系の作成』を参照してください。
- 処理するデータに高さや深さの座標 (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、およびスケール因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。

2. 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、77 ページの『空間参照系の作成』必要があります。

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

- 85 ページの『空間列を作成する』
- 87 ページの『空間列の登録』

関連概念:

- 75 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因子』
- 168 ページの『DB2 Geodetic Extender を使用すべき場合と DB2 Spatial Extender を使用すべき場合』
- 70 ページの『空間参照系』

関連タスク:

- 77 ページの『空間参照系の作成』
- 85 ページの『空間列を作成する』
- 87 ページの『空間列の登録』
- 『空間参照系を作成する: Spatial Extender help』
- 『地理情報列を地理情報基準システムに登録する: Spatial Extender help』
- 『地理情報参照システムを選択する: Spatial Extender help』

関連資料:

- 73 ページの『DB2 Spatial Extender とともに提供される空間参照系』
- 223 ページの『DB2 Geodetic Extender 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

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系、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因数を示しています。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 (DB2 Spatial Extender のサンプル・データ CD に含まれているデータは、すべて小数の度数です) は、デフォルト空間参照系のオフセット値とスケール因数で、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。

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 座標系など、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。どの座標系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。

デフォルトの空間参照系を使用すべきか、新たなシステムを作成すべきかを決定する際には、(71 ページの『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』を参照) 際には、下の表を参考にしてください。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。詳しくは、77 ページの『空間参照系の作成』を参照してください。

表 4. DB2 Spatial Extender で提供されている空間参照系

空間参照系	SRS ID	座標系	オフセット値	スケール因数	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、データが座標系から独立していて、座標系を指定できない、あるいは指定する必要がない場合に選択できます。
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 4. 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	座標系	オフセット値	スケール因数	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系をベースにした空間参照系です。

関連概念:

- 70 ページの『空間参照系』

関連資料:

- 305 ページの『DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビュー』

座標データを整数にトランスフォームする変換因子

DB2[®] Spatial Extender はオフセット 値とスケール 因数を使用して、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因数を持っています。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因数 (場合によってはオフセット値も) を指定する必要があります。詳しくは、77 ページの『空間参照系の作成』を参照してください。

オフセット値

オフセットというのは、剰余として正の値だけが残されるよう、値を調整するために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ります。

プロジェクト用の地理情報リソースのセットアップ

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。たとえば、『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。たとえば、`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
```

スケール因数

スケール因数とは、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。変換によって得られる値が、 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 (たとえば、`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因数は `ディメンション Scale` で表されます (たとえば、`xScale` はすべての X 座標に適用されるスケール因数です)。

```
(max(x) - xOffset) * xScale ≤ 253
(max(y) - yOffset) * yScale ≤ 253
(max(z) - zoffset) * zScale ≤ 253
(max(m) - moffset) * mScale ≤ 253
```

手持ちの座標データに最適のスケール因数を選択する場合、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因数を使用する。
- 小数の X 座標または小数の Y 座標で乗算されたときに、スケール因数が 2^{53} より小さくなるようにする。そのためには、スケール因数を 10 の累乗にするという方法がよく使われます。つまり、スケール因数を、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。
- スケール因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:

たとえば、`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 座標のスケール因数によって、値 10.01 および値 20.03 を乗算します。このスケール因数が 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系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。しかしスケール因数が 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因数の単位

既存の地理情報参照システムを使用するか、新規のシステムを作成するにかかわらず、オフセット値とスケール因数の単位は使用している座標系のタイプによって左右されます。たとえば、地理座標系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。投影座標系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。

関連タスク:

- 71 ページの『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』

空間参照系の作成

DB2 Spatial Extender が提供する空間参照系のどれもが扱うデータを処理できない場合は、新規の空間参照系を作成する必要があります。

手順:

新規の空間参照系を作成するには、次のようにします。

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 (Create Spatial Reference System)」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から **db2se create_srs** を出す。詳しくは、131 ページの『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』を参照してください。
 - db2se.ST_create_srs ストアード・プロシージャを呼び出すアプリケーションを実行する。詳しくは、250 ページの『ST_create_srs』を参照してください。
2. 適切な空間参照系 ID (SRID) を指定する。
 - 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する。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因数を指定する。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。

- DB2 コントロール・センターの「空間参照系の作成 (Create Spatial Reference System)」ウィンドウで、「**エクステント (Extents)**」を選択する。
- **db2se create_srs** コマンド、あるいは db2se.ST_create_srs ストアード・プロシージャに該当のパラメーターを指定する。

プロジェクト用の地理情報リソースのセットアップ

- 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) とスケール因子 (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因子は、以下のいずれかの方法で指定できます。

- DB2 コントロール・センターの「空間参照系の作成 (Create Spatial Reference System)」ウィンドウで、「オフセット (Offset)」を選択する。
- **db2se create_srs** コマンド、あるいは db2se.ST_create_srs ストアド・プロシージャに該当のパラメーターを指定する。

詳しくは、75 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因子』を参照してください。

4. 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を介してこの情報を渡す。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。

- スケール因数。処理する座標の中に小数値が含まれている場合は、スケール因子を算出します (79 ページの『スケール因数の計算』を参照)。スケール因数とは、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のことです。座標が整数である場合は、スケール因数は 1 に設定できます。座標が小数の値の場合は、スケール因数は、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。たとえば、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 のスケール因数が必要になります。

- 座標と指標の最小および最大値。(詳しくは、80 ページの『最小および最大の座標と指標を判別する』を参照してください)。

- 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、剰余として正の値だけが残されるよう、値を調整するために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)。(詳しくは、81 ページの『オフセット値の計算』を参照)。

- スケール因数

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、使用するスケール因数を決定し、そのスケール因数を「空間参照系の作成 (Create Spatial Reference System)」ウィンドウに入力します。詳しくは、79 ページの『スケール因数の計算』を参照。

5. **db2se create_srs** コマンド、あるいは db2se.ST_create_srs ストアド・プロシージャを実行します。

たとえば、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¥"
```

db2se.ST_create_srs ストアード・プロシージャを呼び出すアプリケーションの実行方法の詳細については、250 ページの『ST_create_srs』を参照。

空間参照系が作成できたら、それを、以下のいずれかの作業を行う際に空間列に関連付けます。

- 85 ページの『空間列を作成する』
- 87 ページの『空間列の登録』

関連概念:

- 75 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因子』
- 61 ページの『座標系』
- 70 ページの『空間参照系』

関連タスク:

- 79 ページの『スケール因数の計算』
- 80 ページの『最小および最大の座標と指標を判別する』
- 81 ページの『オフセット値の計算』
- 85 ページの『空間列を作成する』
- 87 ページの『空間列の登録』
- 『空間参照系を作成する: Spatial Extender help』
- 『地理情報列を地理情報基準システムに登録する: Spatial Extender help』

関連資料:

- 131 ページの『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』
- 250 ページの『ST_create_srs』

スケール因数の計算

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因数を算出する必要があります。スケール因数とは、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のことです。

前提条件:

データの処理に使用するスケール因数の算出は、必ず、75 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因子』を選択する際の指針を理解した上で行ってください。

手順:

スケール因数を計算するには、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 (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 の何乗を因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。たとえば、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、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。

スケール因数が算出できたら、次は エクステント値を判別 (『最小および最大の座標と指標を判別する』を参照。) します。そして次に `db2se create_srs` コマンド、あるいは `db2se.ST_create_srs` ストアド・プロシージャを実行します。

関連概念:

- 75 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因子』
- 70 ページの『空間参照系』

関連タスク:

- 80 ページの『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を判別する

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。

前提条件:

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 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ため、新たな空間参照系を作成しようとしている。詳しくは、71 ページの『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』を参照してください。

- 座標の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。

手順:

表そうとしている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。

- X 座標の最小および最大値を判別する。

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。たとえば、油田を表そうとしていて、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。

- Y 座標の最小および最大値を判別する。

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。

- Z 座標の最小および最大値を判別する。

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。

- 最小および最大の指標を判別する。

空間データに指標を組み込む場合は、どの指標が最も高く、どの指標が最も低いかを判別します。

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 to してください。たとえば、最大 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。

判別した最小、最大値の中に小数点以下の桁を含むものがあつた場合は、そのスケール因数を計算する必要があります (79 ページの『スケール因数の計算』を参照)。そうでない場合、`db2se create_srs` コマンド、あるいは `db2se.ST_create_srs` ストアド・プロシージャを実行します。

関連タスク:

- 71 ページの『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』
- 79 ページの『スケール因数の計算』

オフセット値の計算

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、値を調整するために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。

プロジェクト用の地理情報リソースのセットアップ

前提条件:

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。

手順:

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。80 ページの『最小および最大の座標と指標を判別する』を参照。
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。たとえば、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。

たとえば、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。

オフセット値の計算の後には、空間参照系を作成します (77 ページの『空間参照系の作成』を参照)。

関連タスク:

- 80 ページの『最小および最大の座標と指標を判別する』
- 77 ページの『空間参照系の作成』

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、座標系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 GRAPHIC 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
- 列を指定する作業
- 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作業

空間列

表示可能な内容をもった空間列

地理情報列を照会するのに 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

地理情報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。84 ページの図 12 はこの階層を示しています。この図で、インスタンス化できるタイプは背景が白くなっており、インスタンス化できないタイプは背景に陰影が付いています。

インスタンス化できるデータ・タイプは、ST_Point、ST_LineString、ST_Polygon、ST_GeomCollection、ST_MultiPoint、ST_MultiPolygon、ST_MultiLineString です。

インスタンス化できないデータ・タイプは、ST_Geometry、ST_Curve、ST_Surface、ST_MultiSurface、ST_MultiCurve です。

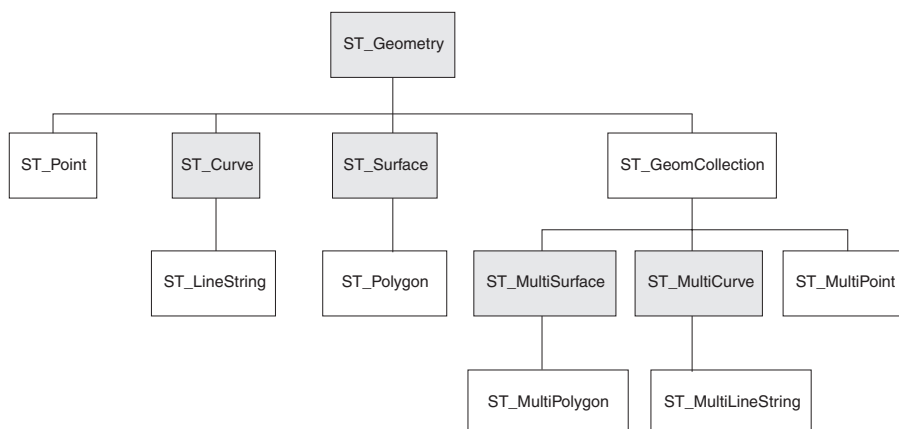


図 12. 空間データの階層： 白いボックスに名前のあるデータ・タイプはインスタンス化できる。

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。たとえば、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。たとえば、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地点を指示します。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ツ (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。たとえば、団地に関する地理情報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。たとえば、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。たとえば、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。たとえば、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、地理情報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。

関連タスク:

- 87 ページの『空間列の登録』
- 85 ページの『空間列を作成する』

空間列を作成する

この作業は、「プロジェクトのための空間情報をセットアップする」という作業の一部です。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するかします。

前提条件:

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。

空間列のセットアップ

- 列を含む表が入っているデータベースに関する SYSADM 権限または 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。
 - 索引の暗黙または明示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
- 表のスキーマに対する ALTERIN 特権

手順:

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用します。

- DB2 の CREATE TABLE ステートメントを使用して表を作成し、その表に地理情報列を組み込む。
- DB2 の ALTER TABLE ステートメントを使用して、地理情報列を既存の表に加える。
- DB2 Control Center 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。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。
- 形状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。91 ページの『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。
- SDE 転送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し、視覚化ツールがその列にアクセスできるようにする。92 ページの『SDE 転送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。

次の作業： 87 ページの『空間列の登録』

関連タスク:

- 91 ページの『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』
- 92 ページの『SDE 転送データを新規の表または既存の表にインポートする』
- 87 ページの『空間列の登録』
- 141 ページの『DB2 Spatial Extender 用のアプリケーションを作成する』
- 『地理情報列を作成する: Spatial Extender help』

関連資料:

- 「SQL リファレンス 第 2 巻」の『ALTER TABLE ステートメント』
- 「SQL リファレンス 第 2 巻」の『CREATE TABLE ステートメント』
- 131 ページの『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』

空間列の登録

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

地理情報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、列のデータの保全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。

- 地理情報索引によるアクセス

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がある空間列中のすべてのデータについて同じ座標系を使用します。空間列は、すべてのデータで必ず同じ空間参照系と、同じ座標系が使用されるよう登録します。

前提条件:

地理情報列を登録するには、ユーザー ID が次のいずれかの権限をもっている必要があります。

- 列を含む表が入ったデータベースに関する SYSADM 権限または DBADM 権限
- この表に対する CONTROL または ALTER 特権。

db2se コマンド行プロセッサやアプリケーション・プログラムを使用して、SDE 転送ファイルからデータをインポートする場合は、DB2 Spatial Extender が、データを保持する列を自動的に作成し登録するようにできます。この場合は、ユーザー ID が、データベースに関して SYSADM 権限か DBADM 権限をもっている必要があります。

手順:

以下のいずれかの方法によって、地理情報列を登録できます。

- DB2 コントロール・センターの「空間列 (Spatial Columns)」と「空間参照系の選択 (Select Spatial Reference System)」のウィンドウを使用して、列を登録する。
- **db2se register_spatial_column** コマンドを出す。
- db2gse.ST_register_spatial_column ストアード・プロシージャを呼び出すアプリケーションを実行する。
- SDE 転送ファイルから空間データをインポートしたい場合は、コントロール・センターの「空間データのインポート (Import Spatial Data)」ウィンドウや **import_sde** コマンド、または db2gse.ST_import_sde ストアード・プロシージャを使用して、地理情報列をもった表を作成し、その列を登録して、データを列にインポートできます。

列を登録した後、DB2GSE.GSE_GEOMETRY_COLUMNS ビューの SRS_NAME 列を参照して、特定の列の選択した空間参照系をチェックしてください。

関連タスク:

- 141 ページの『DB2 Spatial Extender 用のアプリケーションを作成する』

空間列のセットアップ

関連資料:

- 131 ページの『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』
- 283 ページの『ST_register_spatial_column』

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となります。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。この章では、以下の項目を説明しています。

- 空間データをデータベースにインポートする作業、および空間データをアプリケーションが使用できるようなファイルにエクスポートする作業の概要説明。
- ジオコーディングについての説明および、ジオコーディング操作のセットアップ、ジオコーダーを自動的に実行するためのセットアップ、およびジオコーダーをバッチ・モードで実行する作業の紹介。

空間データのインポートおよびエクスポート方法

このセクションでは、データのインポートとエクスポートの概念および、次の作業を紹介します。

- 空間データを、新しい表に、または既存の表またはビューにインポートする
- 空間データを、アプリケーションが使用するファイルにエクスポートする

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、すでに業界で使用可能になっている大量の地理情報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 ユーティリティを使って、データをデータ交換ファイルに書き込みま

空間列にデータを入れる

す。次に、DB2 Spatial Extender を使って、このファイルから、データを地理情報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、地理情報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。

形状ファイルと SDE 転送ファイル:

DB2 Spatial Extender は、形状ファイルと SDE 転送ファイルの 2 種類のデータ交換ファイルをサポートしています。形状ファイルは、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系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。

SDE 転送ファイルは多くの場合、ESRI データベースで生成されるデータをインポートするために使用します。各ファイルには、空間データ、このデータの空間参照系、およびビジネス・データが入っています。ESRI が著作権をもつフォーマットの空間データは、DB2 Spatial Extender カタログに登録されている表の列用のものです。ビジネス・データは、登録済みの列が属する表にあるその他の列用のデータです。

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作業の概要を説明します。このセクションには、これらの作業を実行するために知る必要のある細目（たとえば、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。具体的には、次のことを行えます。

- 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、地理情報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。

前提条件:

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 は次のどれかの権限をもっている必要があります。

- 表またはビューが入ったデータベースに関する SYSADM 権限または DBADM 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)
- 入力ファイルやエラー・ファイルが入るディレクトリへのアクセス権
- 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 が次のどれかの権限をもっている必要があります。

- 表が入ったデータベースに関する SYSADM 権限、DBADM 権限、または CREATETAB 権限
- 次のいずれかの許可:
 - 表が属するスキーマに関する CREATEIN 特権 (スキーマがすでに存在する場合に必要)
 - 表が属するデータベースに関する IMPLICIT_SCHEMA 権限 (表に指定したスキーマが実際には存在しない場合に必要)
- 入力ファイルやエラー・ファイルが入るディレクトリへのアクセス権
- 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

手順:

以下のどれかの方法によって、形状データをインポートできます。

- DB2 コントロール・センターの「形状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** を出す。
- **db2gse.ST_import_shape** ストアード・プロシージャを呼び出すアプリケーションを実行する。

推奨:

DB2 で使用できる機能を利用してインポート処理のパフォーマンスを向上させることができます。たとえば、データを既存の表またはユーザーが作成する表にインポ

空間列にデータを入れる

ートする場合、適切な表作成パラメーターを指定することによって、その表を NOT LOGGED INITIALLY として定義します。

関連概念:

- 89 ページの『空間データのインポートとエクスポートについて』

関連タスク:

- 92 ページの『SDE 転送データを新規の表または既存の表にインポートする』
- 141 ページの『DB2 Spatial Extender 用のアプリケーションを作成する』

関連資料:

- 131 ページの『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』
- 270 ページの『ST_import_shape』

SDE 転送データを新規の表または既存の表にインポートする

SDE 転送データを既存の表にインポートしたり、1 回の操作で、表を作成し、この表に SDE 転送データをインポートしたりできます。具体的には、次のことを行えます。

- DB2 Spatial Extender カタログに登録済みの地理情報列の入った既存の表に、SDE 転送データをインポートする。転送データには、表の地理情報列用の空間データと表のその他の列用のビジネス・データを入れることができます。
- 地理情報列をもった表を自動的に作成し、この列をカタログに登録し、SDE 転送データをこの列と表のその他の列にインポートする。

前提条件:

既存の表やビューの列にデータをインポートするには、ユーザー ID が次のどれかの権限をもっている必要があります。

- 表またはビューが入ったデータベースに関する SYSADM 権限または DBADM 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 特権と SELECT 特権の両方

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操作を開始するには、ユーザー ID が次のどれかの権限を持っている必要があります。

- 表が入ったデータベースに関する SYSADM 権限、DBADM 権限、または CREATETAB 権限
- 次のいずれかの許可:
 - 表が属するスキーマに関する CREATEIN 特権 (スキーマがすでに存在する場合に必要)
 - 表が属するデータベースに関する IMPLICIT_SCHEMA 権限 (表に指定したスキーマが実際には存在しない場合に必要)

手順:

以下のどれかの方法によって、SDE 転送データをインポートできます。

- DB2 コントロール・センターの「インポート (Import)」ウィンドウを使用する。

- **db2se import_sde** コマンドを出す。
- `db2gse.GSE_import_sde` ストアド・プロシージャを呼び出すアプリケーションを実行する。

これらの操作を実行する方法については、この説明の最後にある『関連タスク』に示された項目を参照してください。

関連概念:

- 89 ページの『空間データのインポートとエクスポートについて』

関連タスク:

- 91 ページの『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』
- 141 ページの『DB2 Spatial Extender 用のアプリケーションを作成する』

関連資料:

- 131 ページの『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』
- 240 ページの『GSE_import_sde』

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作業の概要を説明します。このセクションには、これらの作業を実行するために知る必要のある細目 (たとえば、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。このデータは、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。

前提条件:

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。

手順:

空間列にデータを入れる

以下のいずれかの方法で形状ファイルにデータをエクスポートすることができます。

- DB2 コントロール・センターの「形状ファイルへのエクスポート (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から、**db2se export_shape** コマンドを出す。
- db2gse.ST_export_shape ストアード・プロシージャを呼び出すアプリケーションを実行する。

これらの操作を実行する方法については、この説明の最後にある『関連タスク』に示された項目を参照してください。

データを SDE 転送ファイルにエクスポートする

空間データが入っている表を SDE 転送ファイルにエクスポートできます。この表には、地理情報列を複数、入れることはできません。さらに、この列は、DB2 Spatial Extender カタログに登録しておく必要があります。表にビジネス・データが入っている場合は、そのデータは空間データとともにエクスポートされます。表の中のすべての行をエクスポートすることも、行のサブセットをエクスポートすることもできます。サブセットをエクスポートするには、サブセットを指定する WHERE 文節を使用します。

前提条件:

SDE 転送ファイルにデータをエクスポートするには、ユーザー ID に次の権限が必要です。

- SYSADM または DBADM 権限
- エクスポートされる表に対する SELECT 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

制約事項:

- 各エクスポート操作でエクスポートできる地理情報列は 1 つだけである。
- エクスポートする列のデータ・タイプは、SDE フォーマットがサポートするデータ・タイプでなければならない。
- この表に含まれる地理情報列は、ただ 1 つである必要がある。
- この列は DB2 Spatial Extender カタログに登録しておく必要がある。
- 既存の SDE ファイルに付加することはできません。

手順:

以下のどれかの方法で、空間データとビジネス・データを SDE 転送ファイルにエクスポートすることができます。

- DB2 コントロール・センターの「SDE ファイルへのエクスポート (Export SDE Files)」ウィンドウを使用する。
- **db2se export_sde** コマンドを出す。
- db2gse.GSE_export_sde ストアード・プロシージャを呼び出すアプリケーションを実行する。

関連概念:

- 89 ページの『空間データのインポートとエクスポートについて』

関連タスク:

- 141 ページの『DB2 Spatial Extender 用のアプリケーションを作成する』

関連資料:

- 131 ページの『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』
- 238 ページの『GSE_export_sde』

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作業を紹介します。

- ジオコーダーに発行させようとする作業を定義する。たとえば、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作業の要約を行います。

DB2[®] Spatial Extender では、ジオコーダーは、既存データ (関数の入力) を地理情報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。たとえば、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。たとえば、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 (アドレスを座標に変換する) 操作のことを意味します。

空間列にデータを入れる

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。たとえば、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 トリガーによって活動化されます。

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 表にある地理情報列の説明を導き出すいくつかの操作のうち 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。たとえば、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地理情報列の中に入れます。
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。

ユーザーの作業:

DB2 Spatial Extender では、ジオコーディングに関する作業は次のとおりです。

- 指定する地理情報列について、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。たとえば、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、何個のレコードを処理するかを決定すること、などです。この作業は、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)。この作業は、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。ユーザーがすで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。この作業は、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。たとえば、次のことを指定できます。

- 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数。
- ジオコーダーに固有の操作に関する要件。たとえば、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、前述した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。たとえば、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。

前提条件:

空間列にデータを入れる

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 SYSADM 権限または DBADM 権限。
- ジオコーダーが操作する各表に関する SELECT 特権、および CONTROL 特権または UPDATE 特権。

手順:

以下のどれかの方法によって、ジオコーディング操作をセットアップできます。

- 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 (Set Up Geocoding)」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。

これらの操作を実行する方法については、この説明の最後にある『関連タスク』に示された項目を参照してください。

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、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、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がある最小度合いは、**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がある最小度合いは、**最小一致スコア** と呼ばれます。たとえば、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ぬ場合もあるので注意してください。たとえば、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。たとえば、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。たとえば、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可能性があります。

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。

前提条件:

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地理情報列ごとに、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、SYSADM 権限または DBADM 権限。
 - 以下に示すような、この表に関する 1 つ以上の特権。
 - CONTROL 特権。
 - CONTROL 特権をもたない場合は、ALTER 特権、SELECT 特権、および UPDATE 特権が必要です。
 - この表でトリガーを作成するのに必要な特権。

手順:

自動ジオコーディングをセットアップするには、3 通りの方法があります。

- 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 (Set Up Geocoding)」ウィンドウ、または「ジオコーディング (Geocoding)」ウィンドウからセットアップする。
- **db2se enable_autogc** コマンドを出す。

空間列にデータを入れる

- db2gse.ST_enable_autogeocoding ストアド・プロシージャを呼び出すアプリケーションを実行する。

これらの操作を実行する方法については、この説明の最後にある『関連タスク』に示された項目を参照してください。

推奨事項:

- 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地理情報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地理情報列のデータよりも、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すで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。

バッチ・モードでのジオコーダーの実行

バッチ・モードで実行されるジオコーダーを呼び出すことができます。このジオコーダーは、1 回の操作で、複数のレコードを、特定の列に入る空間データに変換しようとしています。

特定の地理情報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。たとえば、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。

前提条件:

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 SYSADM 権限または DBADM 権限
- この表に関する CONTROL 特権または 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 文節を指定するときは、これらの文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。

制約事項:**手順:**

以下のどれかの方法によって、バッチ・モードで実行されるジオコーダーを呼び出すことができます。

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

空間列にデータを入れる

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。たとえば、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格子索引を利用して、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格子索引を提供しています。

- 測地ポロノイ・インデックス

DB2 Geodetic Extender 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操作手法をサポートしています。測地ポロノイ索引は、地球を連続した球体として扱い、極地や、第 180 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。

関連概念:

- 185 ページの『測地ポロノイ・インデックス』
- 104 ページの『空間グリッド・インデックス』

関連タスク:

- 189 ページの『測地ポロノイ・インデックスの作成』
- 111 ページの『地理情報格子索引の作成』

関連資料:

- 128 ページの『照会を最適化するために索引を使用する関数』

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、処理対象となる行を大幅に減らすことができます。

DB2 Spatial Extender は、2 ディメンション・データ向けに最適化された格子索引を提供しています。索引は図形の 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、格子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、図形の最小外接長方形 (MBR) を使用して生成します。ほとんどの図形の場合、MBR はその図形を囲む長方形です。MBR の詳細については、442 ページの『ST_MBR』を参照してください。

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格子に分割します。格子セルと各図形の 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、地理情報索引が地理情報列に対して作成されます。索引の各項目は、格子セルの ID、図形の MBR、図形を含む行の内部 ID から構成されます。

空間インデックス・レベル (格子レベル) は最大で 3 つ定義できます。複数の格子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。詳細については、106 ページの『索引レベル数と格子サイズに関する考慮事項』を参照してください。

4 つ以上の格子セルと交差する図形は、次のより大きなレベルに格上げされます。一般に、大きな図形は、大きなサイズのレベルで索引化されます。図形が、大きな格子サイズでも 10 以上の格子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格子サイズを定義する必要があります。

複数の格子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、可能な限り低い格子レベルを使おうとします。あるレベルの図形が 4 つ以上の格子セルと交差する場合は、次に大きいレベルに昇格します (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 つの格子レベルのある空間インデックスは、まず、レベル 10.0 の格子と各図形を交差させます。図形が 4 つ以上の 10.0 サイズの格子セルと交差する場合、昇格してレベル 100.0 の格子と交差するようになります。4 つ以上の交差が 100.0 レベルで生じている

と、この図形は 1000.0 レベルへ昇格します。10 以上の交差が 1000.0 レベルで生じている場合には、図形はオーバーフロー・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 UDB オプティマイザーは、照会の WHERE 文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

詳細については、128 ページの『照会を最適化するために索引を使用する関数』を参照してください。

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、照会の実行時に以下の複数ステップのフィルタリング処理が行われます。

1. どの格子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、処理対象となる図形のこと、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、格子セル ID が合致する項目を検索する。
3. 索引項目中の図形の 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られた図形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) を定義するものです。

2 つ目の例では、Spatial Extender はホスト値に指定された図形 (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格子サイズを指定すべきです。格子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格子セル中に存在する、という図形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、格子サイズが小さすぎると、図形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格子サイズ提案する索引アドバイザー・ユーティリティを提供しています。詳しくは、116 ページの『空間グリッド・インデックス作成のための格子サイズの決定』を参照してください。

関連概念:

- 106 ページの『索引レベル数と格子サイズに関する考慮事項』
- 103 ページの『空間インデックスのタイプ』
- 115 ページの『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』

関連タスク:

- 111 ページの『地理情報格子索引の作成』

関連資料:

- 114 ページの『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』

索引レベル数と格子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格子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。このセクションでは、格子レベルや格子サイズを変えることによってどのような効果があるかをおおまかに解説します。

格子レベルの数

格子レベルは 3 つまで持つことができます。しかし、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格子レベルについて個別に索引が検索されます。したがって、格子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、格子レベルを 1 とする。通常、空間列内に同じ相対サイズの図形が複数含まれることはありませんが、空間列中の図形をサイズによってグループ化することは可能です。格子レベルは、この図形グループに対応させる必要があります。

たとえば、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 つの格子レベルを指定することになります。

格子セル・サイズ

索引項目数を最少にしながら、格子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。

- 列内の小さな図形に対する索引全体を最適化するには、最小格子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、検索領域内にはない図形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、最小格子サイズでは索引入力も最大になります。したがって、照会時に処理される索引入力数が増加すると、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。
- 大きな格子サイズを使用すると、大きな図形に対する索引の最適化をさらに進めることができます。大きな図形に大きな格子サイズを使用すると、最小格子サイズで生成されるよりも索引入力は少なくなります。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、格子サイズを変えることによる効果がわかります。

108 ページの図 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。たとえば、MBR が照会領域と交差している図形をすべて検索したいとします。108 ページの図 13 を見ると、28 の図形 (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。

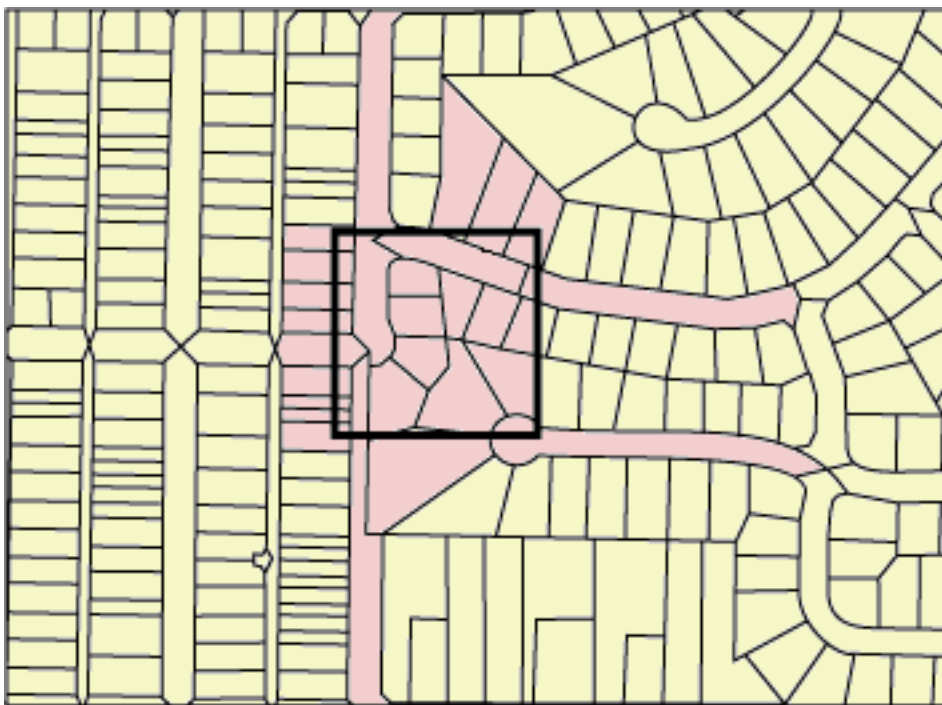


図 13. 近隣の土地の区画

109 ページの図 14 は、格子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 の図形のみを戻しますが、 MBR が照会領域と交差している 3 つの余分な図形を調べ、その後で破棄するという処理が必要になります。

- 格子サイズを小さくすることで、図形 1 つあたりの索引項目は増えます。照会は実行時に、31 の図形のすべての索引項目にアクセスします。109 ページの図 14 は、256 の格子セルが、照会領域に重なっている状態を示しています。この場合、多くの図形が複数の同じ格子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、格子サイズを小さくすると、余分な索引項目へのアクセスが増えます。

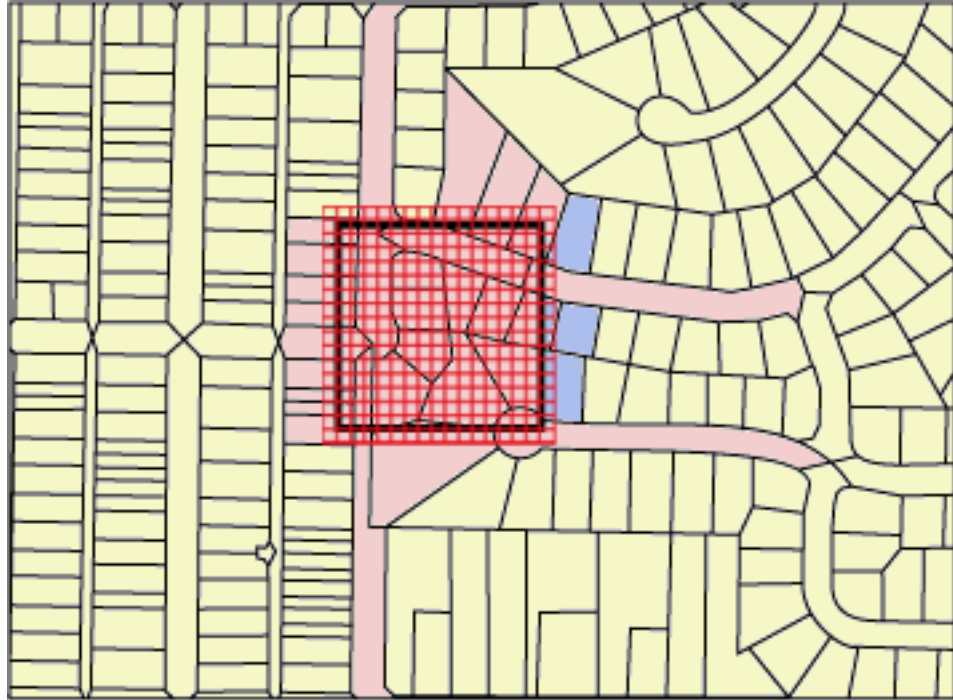


図 14. 土地区画の格子サイズを小さく (25 に) した場合

110 ページの図 15 は、格子サイズを大きく (400 に) し、照会領域より広い範囲の図形に重なるようにした例です。

- このように格子サイズを大きくすると、各図形に対応する索引項目は 1 つのみになりますが、照会は、MBR が格子セルと交差する 59 もの余分な図形について調べ、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 の図形のすべての索引項目にアクセスし、さらに、59 の余分な図形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、格子サイズを大きくすると、数多くの余分な図形へのアクセスが必要になります。

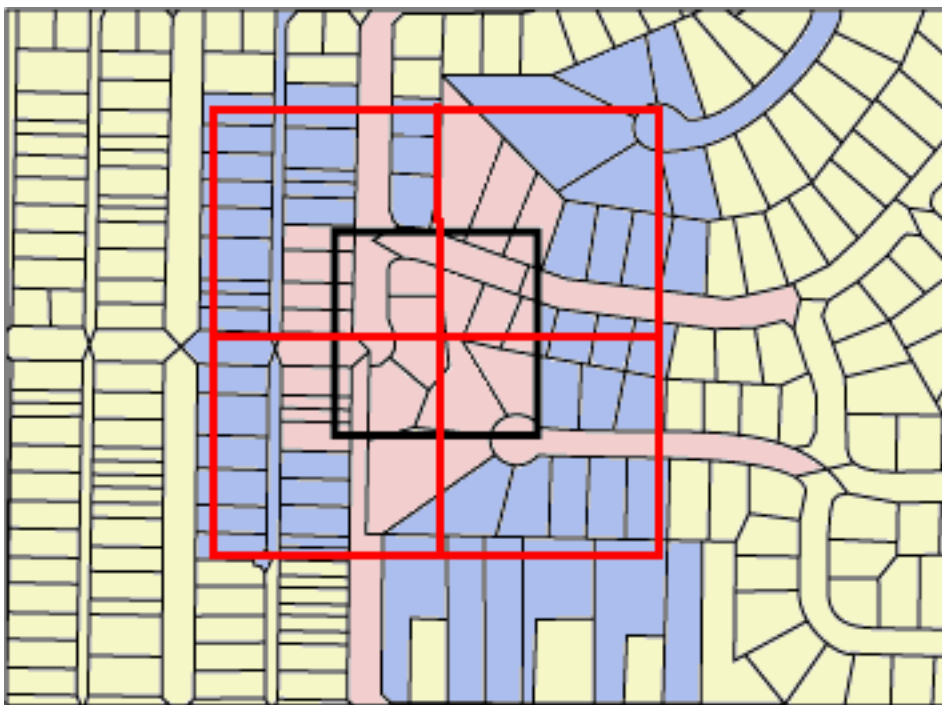


図 15. 格子サイズを大きく (400 に) した場合

111 ページの図 16 は、格子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 の図形のみを戻しますが、MBR が照会領域と交差している 5 つの余分な図形を調べ、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 の図形のすべての索引項目にアクセスし、さらに、5 つの余分な図形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、中程度の格子サイズが最適ということになります。格子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、格子サイズが大きい場合に比べ、余分な図形にアクセスすることも減るからです。

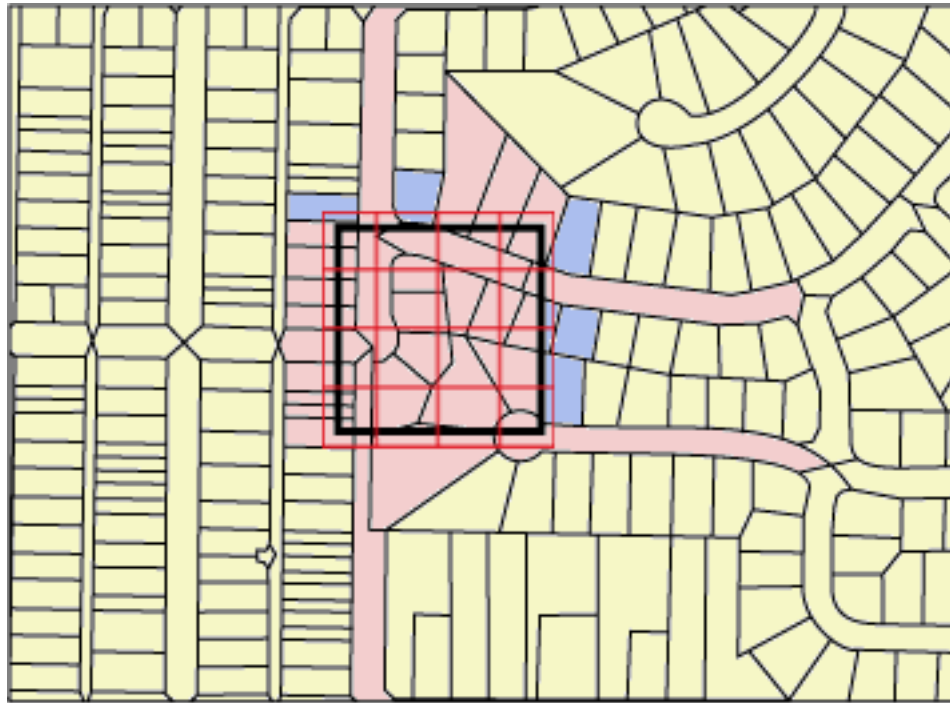


図 16. 格子サイズを中程度に (100 に) した場合

関連概念:

- 104 ページの『空間グリッド・インデックス』
- 115 ページの『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』

関連タスク:

- 116 ページの『空間グリッド・インデックス作成のための格子サイズの決定』
- 118 ページの『空間グリッド・インデックスに関する統計の分析』
- 111 ページの『地理情報格子索引の作成』

関連資料:

- 128 ページの『照会を最適化するために索引を使用する関数』

地理情報格子索引の作成

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
- 格子サイズ (104 ページの『空間グリッド・インデックス』を参照)。3 つの格子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。

前提条件:

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 SYSADM 権限または DBADM 権限
 - 以下に示す権限、特権の両方。
 - 以下の特権のいずれか。
 - 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。
 - 索引の暗黙または明示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 つの格子のサイズが分かっている。詳しくは、116 ページの『空間グリッド・インデックス作成のための格子サイズの決定』を参照。

推奨事項:

- 列に関する地理情報格子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、地理情報列データを分析し、その地理情報格子索引に適切な格子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格子セル・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 UDB は、処理中に別名を解決します。

手順:

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「空間エクステンダー (Spatial Extender)」ウィンドウを使用する。
- EXTEND USING 文節で 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。

このセクションでは、最初の 2 つの方法のためのステップを示します。地理情報格子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。

コントロール・センターを使用して地理情報格子索引を作成するには、以下のようになります。

1. コントロール・センターで、地理情報格子索引を作成しようとする地理情報列が入っている表を右クリックし、ポップアップ・メニューから「**Spatial Extender**」→「**地理情報索引 (Spatial Indexes)**」を選択する。「空間インデックス」ウィンドウが開きます。
2. 「空間インデックス」ウィンドウのオンライン・ヘルプの説明に従う。空間インデックス (Spatial Index) ウィンドウの「ヘルプ (Help)」 ボタンをクリックすると、これらの説明を表示できます。

SQL CREATE INDEX ステートメントを使用してこの作業を行う方法：

1. EXTEND USING 文節と db2gse.spatial_index 格子索引拡張子を使用して CREATE INDEX ステートメントを決定する。

たとえば、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・ライン・プロセッサで CREATE INDEX コマンドを発行する。

関連概念:

- 104 ページの『空間グリッド・インデックス』
- 106 ページの『索引レベル数と格子サイズに関する考慮事項』
- 115 ページの『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』

関連タスク:

- 116 ページの『空間グリッド・インデックス作成のための格子サイズの決定』
- 118 ページの『空間グリッド・インデックスに関する統計の分析』

関連資料:

- 「SQL リファレンス 第 2 巻」の『CREATE INDEX ステートメント』
- 114 ページの『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』
- 128 ページの『照会を最適化するために索引を使用する関数』

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 文節を指定して使用してください。

構文:

```

▶ CREATE INDEX index_schema. index_name ON
  table_schema. table_name (column_name) EXTEND USING
  db2gse.spatial_index (finest_grid_size, middle_grid_size,
  coarsest_grid_size)

```

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 UDB 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

作成する格子索引の非修飾名。

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格子サイズ。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 でなければならぬ。
- *coarsest_grid_size* は *middle_grid_size* より大きいか、0 でなければならぬ。

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、最初の図形の索引作成時に格子サイズの有効性がチェックされます。したがって、指定した格子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。

- 空間グリッドのすべて図形が NULL であれば、Spatial Extender は格子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 でない図形が挿入されるか、空間列中の NULL でない図形が更新される場合に、格子サイズの妥当性を確認します。指定された格子サイズが無効である場合は、非 NULL の図形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 の図形が存在した場合、Spatial Extender はその時点で格子サイズの妥当性を確認します。指定された格子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)
```

関連概念:

- 106 ページの『索引レベル数と格子サイズに関する考慮事項』
- 104 ページの『空間グリッド・インデックス』
- 115 ページの『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』

関連タスク:

- 116 ページの『空間グリッド・インデックス作成のための格子サイズの決定』
- 118 ページの『空間グリッド・インデックスに関する統計の分析』
- 111 ページの『地理情報格子索引の作成』

関連資料:

- 128 ページの『照会を最適化するために索引を使用する関数』

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格子サイズを判別する。

索引アドバイザーは、空間列中の図形を分析し、空間グリッド・インデックスに最適な格子サイズを推奨します。その詳しい手順については、116 ページの『空間グリッド・インデックス作成のための格子サイズの決定』を参照してください。

- 既存の格子索引を分析する。

索引アドバイザーは、現在の格子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。その手順については、118 ページの『空間グリッド・インデックスに関する統計の分析』を参照してください。

関連概念:

- 106 ページの『索引レベル数と格子サイズに関する考慮事項』
- 104 ページの『空間グリッド・インデックス』

関連タスク:

- 116 ページの『空間グリッド・インデックス作成のための格子サイズの決定』
- 118 ページの『空間グリッド・インデックスに関する統計の分析』

関連資料:

- 114 ページの『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』
- 123 ページの『gseidx コマンド』
- 128 ページの『照会を最適化するために索引を使用する関数』

空間グリッド・インデックス作成のための格子サイズの決定

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、適切な格子サイズを判別できます。

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があります。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。たとえば以下の 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クールを作成し、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クールを作成することもできます。

手順:

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

- 作成する索引に対する推奨格子セル・サイズを、索引アドバイザーを使って求める。

- 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、格子セル・サイズを要求します。たとえば、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、推奨格子セル・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、たとえば次のような **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

- 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。たとえば、地図の画面の幅が約 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。この行の格子サイズは、1.4、3.5、14.0 となっています。

- 推奨された格子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);
```

関連概念:

- 106 ページの『索引レベル数と格子サイズに関する考慮事項』
- 104 ページの『空間グリッド・インデックス』

関連タスク:

- 118 ページの『空間グリッド・インデックスに関する統計の分析』

関連資料:

- 114 ページの『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』

- 123 ページの『gseidx コマンド』
- 128 ページの『照会を最適化するために索引を使用する関数』

空間グリッド・インデックスに関する統計の分析

既存の空間格子索引に関する統計を見れば、その格子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。

推奨事項: 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・ライン・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。

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があります。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。たとえば、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クラスタを作成し、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クラスタを作成することもできます。

手順:

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. 既存の索引の格子セル・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 文節と 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。たとえば、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** 文節を指定する。DETAIL キーワードを含めます。たとえば、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。たとえば 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. 既存の索引の格子セル・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。

- 図形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、格子サイズを調整します。

索引アドバイザーに 『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 について得られた統計を見ると、以下のような理由から、格子サイズ (0.5, 0, 0) が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、推奨されている 4 を上回っている。

『Index Entries』 行の 1、2、3、4、10 という値は、図形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図形の数を示しています。たとえば、前のステップの出力を見ると、1519 の図形が 4 つの索引項目を持っていることがわかります。索引項目数 10 に対応する 『Absolute』 値は 695 なので、695 の図形が 5 から 10 の間の索引項目を持っていることとなります。

- 『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 の図形が 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. 図形の数が高くなる MBR サイズを探します。『Histogram』 セクションには、MBR サイズと図形数の対応表が表示されます。以下の例では、最高の図形数 (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、推奨格子サイズが (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. この推奨格子サイズが、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、推奨格子サイズを指定して **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、推奨格子サイズはガイドラインに合っていることがわかりません。

- 『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、推奨された格子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);

```

関連概念:

- 106 ページの『索引レベル数と格子サイズに関する考慮事項』
- 104 ページの『空間グリッド・インデックス』

関連タスク:

- 116 ページの『空間グリッド・インデックス作成のための格子サイズの決定』
- 111 ページの『地理情報格子索引の作成』

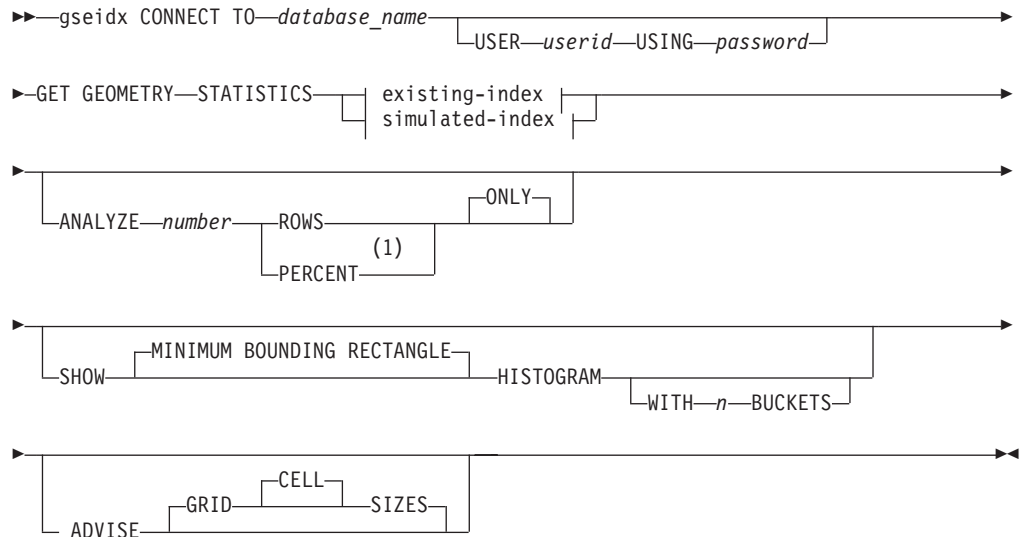
関連資料:

- 114 ページの『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』
- 123 ページの『gseidx コマンド』
- 128 ページの『照会を最適化するために索引を使用する関数』

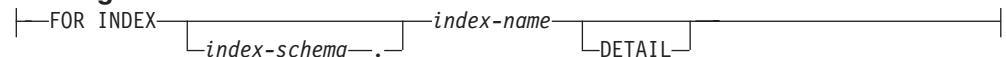
gseidx コマンド

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。

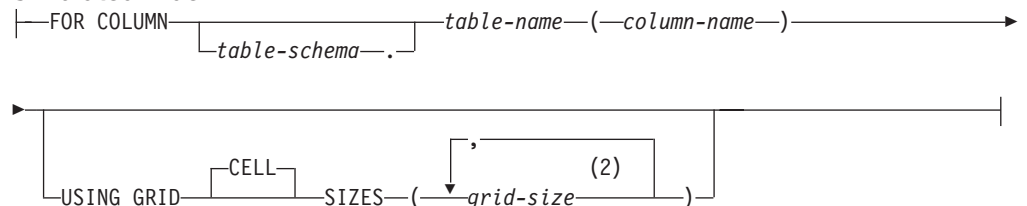
構文



existing-index:



simulated-index:



注:

- 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。
- 1、2、または 3 格子レベルにセル・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 SYSADM 権限か DBADM 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

各格子レベルについて以下の情報を示します。

- 格子セルのサイズ
- 索引が付いた図形の数
- 索引項目数
- 図形を含んでいる格子セルの数
- 図形当たりの平均索引項目数
- 格子セル当たりの平均図形数
- 最も多く図形を含んでいるセル内の図形の数
- 含まれている図形が最も少ないセル内の図形の数

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格子レベル (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、レベルに対して格子セル・サイズを指定しないか、またはレベルに格子セル・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 文節に 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

図形の最小外接長方形 (MBR) のサイズ、および MBR が同じサイズである図形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図形の MBR に関するグループ化の数を指定する。小さい MBR はその他の小さい図形とまとめてグループ化されます。大きい MBR は、他の大きい図形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、たとえば 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。たとえば、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格子セル・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** である既存の格子索引に関する詳細情報を戻し、適切な格子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
```

このコマンドが戻す情報の詳細については、118 ページの『空間グリッド・インデックスに関する統計の分析』を参照してください。

関連概念:

- 106 ページの『索引レベル数と格子サイズに関する考慮事項』
- 104 ページの『空間グリッド・インデックス』
- 115 ページの『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』

関連タスク:

- 116 ページの『空間グリッド・インデックス作成のための格子サイズの決定』
- 111 ページの『地理情報格子索引の作成』

関連資料:

- 114 ページの『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』
- 128 ページの『照会を最適化するために索引を使用する関数』

空間列にアクセスするのにビューを使用する

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、地理情報列を使用するビューを定義できます。地理情報列を持つ表があり、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。

関連タスク:

- 「管理ガイド: インプリメンテーション」の『ビューの作成』

関連資料:

- 「SQL リファレンス 第 2 巻」の『CREATE VIEW ステートメント』

第 12 章 地理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
- 空間処理関数を地理情報索引と組み合わせて使用する場合のガイドライン

地理情報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、地理情報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。一般的には、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 5 は、それらのカテゴリーを例とともに示したものです。表 5 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 5. 空間処理関数および操作

関数のカテゴリー	操作の例
特定の図形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。
既存の図形から新しい図形を派生させる。	そのロケーションから店の販売地域を導き出す。
図形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図形に変換し、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: 特定の図形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系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
```

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 sales_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、顧客の家を表す図形の座標を、Store 10 の販売地域を表す図形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: 既存の図形から新しい図形を派生させる:

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図形から、商店の販売地域を表す図形が引き出されます。

```
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図形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図形に変換して、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、あるいは測地ポロノイ・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 つの図形の比較を行います。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。

たとえば、関数 `ST_Overlaps` は、同じディメンション (たとえば、2 つの折れ線または 2 つのポリゴン) をもつ 2 つの図形を比較します。2 つの図形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図形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。

129 ページの表 6 は、個々の比較関数が、空間グリッド・インデックス、測地ポロノイ・インデックスを使用できるか否かをまとめたものです。

表 6. 比較関数による空間グリッド・インデックス、測地ポロノイ・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	測地ポロノイ・インデックスを使用できるか
EnvelopesIntersect	できる	できる
ST_Contains	できる	できる
ST_Crosses	できる	できない
ST_Distance	できる	できる
ST_EnvIntersects	できる	できる
ST_Equals	できる	できない
ST_Intersects	できる	できる
ST_MBRIntersects	できる	できる
ST_Overlaps	できる	できない
ST_Touches	できる	できない
ST_Within	できる	できる

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、比較する図形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、図形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 文節に指定する必要がある。SELECT、HAVING または GROUP BY 文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。

たとえば、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
```

130 ページの表 7 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 7. 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
<pre>SELECT * FROM stores AS s WHERE db2gse.ST_Contains(s.sales_zone, ST_Point(-121.8,37.3, 1)) = 1</pre>	この例は、どの条件にも違反していない。
<pre>SELECT * FROM stores AS s WHERE db2gse.ST_Length(s.location) > 10</pre>	空間処理関数 ST_Length は、図形を比較せず、空間インデックスを使用できない。
<pre>SELECT * FROM stores AS s WHERE 1=db2gse.ST_Within(s.location,:BayArea)</pre>	関数は、述部の左側の式でなければならぬ。
<pre>SELECT * FROM stores AS s WHERE db2gse.ST_Contains(s.sales_zone, ST_Point(-121.8,37.3, 1)) <> 0</pre>	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。
<pre>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</pre>	この関数のどちらの引き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。

関連概念:

- 106 ページの『索引レベル数と格子サイズに関する考慮事項』
- 185 ページの『測地ポロノイ・インデックス』
- 104 ページの『空間グリッド・インデックス』
- 115 ページの『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』

関連タスク:

- 189 ページの『測地ポロノイ・インデックスの作成』
- 111 ページの『地理情報格子索引の作成』

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 (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できます。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・プロシージャが記載してある表 8 を参照してください。たとえば、**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・プロシージャを呼び出しません。その代わりに、形状ファイルの説明に関する情報を表示します。

手順:

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。

コマンド

以前のバージョンでは、Spatial Extender の各サブコマンドの前には、db2se ではなく gseadm を置く必要がありました。以前のバージョンで作成した gseadm スクリプトはバージョン 8.1 でも動作しますが、db2se コマンド行プロセッサを使用するスクリプトにマイグレーションすることをお勧めします。

次の点に注意してください。

- 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。たとえば、自分とは異なるユーザーとして、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。

注：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。たとえば、次のように指定します。

```
¥"mytable¥"
```

大文字小文字の区別のある値が、別の大文字小文字の区別のある値によって修飾されている場合は、2 つの値を個々の値に区切ります。たとえば、次のようにします。

```
"myschema"."mytable"
```

ストリングは、二重引用符で囲みます。たとえば、次のようにします。

```
"select * from newtable"
```

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・プロシージャは、同じパラメーターを共有します。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 8. 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	<p>db2se create_cs</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_create_coordsys ストアド・プロシージャーのものと同じです。</p> <p>次の例では、「mycoordsys」という名前の座標システムを作成します。</p> <pre>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]]</pre>
空間参照系の作成	<p>db2se create_srs</p> <p>コマンド固有のパラメーターは、db2gse.ST_create_srs ストアド・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。</p> <p>次の例では、「mysrs」という名前の空間参照系を作成します。</p> <pre>db2se create_srs mydb -srsName ¥"mysrs¥" -srsID 100 -xScale 10 -coordsysName ¥"GCS_North_American_1983¥"</pre>
空間参照系のドロップ	<p>db2se drop_srs</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_drop_srsdb2gse.ST_drop_srs ストアド・プロシージャーのものと同じです。</p> <p>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。</p> <pre>db2se drop_srs mydb -srsName ¥"mysrs¥"</pre>
座標システム定義の削除	<p>db2se drop_cs</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_drop_coordsysdb2gse.ST_drop_coordsys ストアド・プロシージャーのものと同じです。</p> <p>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。</p> <pre>db2se drop_cs mydb -coordsysName ¥"mycoordsys¥"</pre>
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	<p>db2se disable_autogc</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_disable_autogeocoding ストアド・プロシージャーのものと同じです。</p> <p>次の例では、表 MYTABLE の MYCOLUMN という名前のジオコード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。</p> <pre>db2se disable_autogc mydb -tableName ¥"mytable¥" -columnName ¥"mycolumn¥"</pre>

1

表 8. タスクに対応した db2se コマンド (続き)

タスク	コマンドと例
地理情報操作作用にデータベースを使用可能にする	<p>db2se enable_db</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_enable_dbdb2gse.ST_enable_db ストアード・プロシージャーのものと同じです。</p> <p>次の例では、地理情報操作作用に、MYDB という名前のデータベースを使用可能にします。</p> <pre>db2se enable_db mydb</pre>
SDE 転送ファイルへのデータのエクスポー ト	<p>db2se export_sde</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.GSE_export_sde db2gse.GSE_export_sde ストアード・プロシージャーのものと同じです。</p> <p>次の例では、地理情報列 MYSPATIALCOLUMN が入った表 MYSDTABLE から、 mysdefile という名前の SDE 転送ファイルにデータをエクスポー トします。</p> <pre>db2se export_sde mydb -tableName ¥"mySDEtable¥" -columnName ¥"mySpatialcolumn¥" -fileName /home/myaccount/mysdefile</pre> <p>次の例では、 SPATIALTABLE という名前の表から、sdex という名前の SDE ファイルにデータをエクスポー トします。このファイルは DB2 クライアントに作成されます。エラーおよび情報メッセージ (たとえばエクスポー トの開始時刻と終了時刻やエクスポー トされた行の数) は、 sdex.export.log というファイルに書き込まれます。</p> <pre>db2se export_sde mydb -client -fileName sdex -selectStatement "SELECT * FROM spatialTable" -messagesFile sdex.export.log</pre>
形状ファイルへのデー タのエクスポー ト	<p>db2se export_shape</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_export_shape ストアード・プロシージャーのものと同じです。</p> <p>次の例では、MYCOLUMN と名付けられた地理情報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。</p> <pre>db2se export_shape mydb -fileName /home/myaccount/myshapefile -selectStatement "select * from mytable"</pre>

表 8. タスクに対応した db2se コマンド (続き)

タスク	コマンドと例
SDE 転送ファイルの インポート	<p>db2se import_sde</p> <p>コマンド固有のパラメーターと必要な権限は、 db2gse.GSE_import_sdedb2gse.GSE_import_sde ストアード・プロシ ジャーのものと同じです。</p> <p>次の例では、mysdefile という名前の SDE 転送ファイルが、 MYSPATIALCOLUMN という名前の地理情報列が入った表 MYSDTABLE にインポートされます。 10 レコードごとに、コミ ットが出されます。</p> <pre>db2se import_sde mydb -tableName ¥"mysdetable¥" -columnName ¥"mySpatialcolumn¥" -fileName /home/myaccount/"mysdefile" -commitScope 10</pre> <p>次の例は、DB2 クライアント上にある sdex という名前の SDE フ ァイルをインポートする方法を示したものです。この例では、デー タは SDETABLE という名前の表 (の ID という名前の列) にインポ ートされ、 100 レコードごとにコミットが出されます。エラーは sdex.exceptions というファイルに書き込まれます。</p> <pre>db2se import_sde mydb -client -filename sdex -srsId 1234 -tableName sdeTable -idColumn id -commitScope 100 -messagesFile sdex.exceptions</pre>
形状ファイルのインポ ート	<p>db2se import_shape</p> <p>コマンド固有のパラメーターと必要な権限は、 db2gse.ST_import_shape ストアード・プロシジャーのものと同じで す。</p> <p>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。</p> <pre>db2se import_shape mydb -fileName ¥"myfile¥" -srsName NAD83_SRS_1 -tableName ¥"mytable¥" -spatialColumnName ¥"mycolumn¥"</pre>
ジオコーダーの登録	<p>db2se register_gc</p> <p>コマンド固有のパラメーターと必要な権限は、 db2gse.ST_register_geocoder ストアード・プロシジャーのものと同 じです。</p> <p>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。</p> <pre>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"</pre>

表 8. タスクに対応した db2se コマンド (続き)

タスク	コマンドと例
地理情報列の登録	<p>db2se register_spatial_column</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_register_spatial_column ストアド・プロシージャーのものと同じです。</p> <p>次の例では、表 MYTABLE の MYCOLUMN という名前の地理情報列を、空間参照系「USA_SRS_1」に登録します。</p> <pre>db2se register_spatial_column mydb -tableName ¥"mytable¥" -columnName ¥"mycolumn¥" -srsName USA_SRS_1</pre>
データベースを地理情報操作に使用できるようにするリソースの除去	<p>db2se disable_db</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_disable_dbdb2gse.ST_disable_db ストアド・プロシージャーのものと同じです。</p> <p>次の例では、地理情報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。</p> <pre>db2se disable_db mydb</pre>
ジオコーディング操作作用セットアップの除去	<p>db2se remove_gc_setup</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_remove_gc_setup ストアド・プロシージャーのものと同じです。</p> <p>次の例では、表 MYTABLE 中の MYCOLUMN という名前の地理情報列に適用されるジオコーディング操作作用のセットアップを除去します。</p> <pre>db2se remove_geocoding_setup mydb -tableName ¥"mytable¥" -columnName ¥"mycolumn¥"</pre>
バッチ・モードでのジオコーダーの実行	<p>db2se run_gc</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_run_gc ストアド・プロシージャーのものと同じです。</p> <p>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。</p> <pre>db2se run_gc mydb -tableName ¥"mytable¥" -ccolumnName ¥"mycolumn¥"</pre>
自動的に実行されるジオコーダーのセットアップ	<p>db2se enable_autogeocoding</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_enable_autogeocoding ストアド・プロシージャーのものと同じです。</p> <p>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。</p> <pre>db2se enable_autogeocoding mydb -tableName ¥"mytable¥" -columnName ¥"mycolumn¥"</pre>

表 8. タスクに対応した db2se コマンド (続き)

タスク	コマンドと例
ジオコーディング操作 のセットアップ	<p>db2se setup_gc</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_setup_geocoding ストアド・プロシージャーのものと同じです。</p> <p>次の例では、表 MYTABLE の MYCOLUMN という名前の地理情報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。</p> <pre>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</pre>
形状ファイルとその説明に関する情報の表示	<p>db2se shape_info</p> <p>このコマンドを使用するには、以下のことが必要です。</p> <ul style="list-style-type: none"> • 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系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)。 <p>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。</p> <pre>db2se shape_info -fileName myfile</pre> <p>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系および空間参照系が、すべて検出されます。</p> <pre>db2se shape_info -fileName ~/sqllib/samples/spatial/data/offices -database myDB</pre>

表 8. タスクに対応した db2se コマンド (続き)

タスク	コマンドと例
SDE ファイルとその説明に関する情報の表示	<p>db2se sde_info</p> <p>このコマンドを使用するには、以下のことが必要です。</p> <ul style="list-style-type: none"> • 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系および空間参照系をシステムが検索するように指定する場合)。 <p>次の例では、現行ディレクトリーにある sdefile という名前の SDE ファイルに関する情報を表示します。</p> <pre>db2se sde_info -fileName myfile</pre> <p>次の例では、sdex という名前の SDE ファイルに関する情報を表示し、MYDB という名前のデータベースで、互換性のある座標系および空間参照系をすべて検索します。</p> <pre>db2se sde_info -fileName data/sdex -database myDB</pre>
ジオコーダーの登録抹消	<p>db2se unregister_gc</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_unregister_geocoder ストアード・プロシージャーのものと同じです。</p> <p>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。</p> <pre>db2se unregister_gc mydb -geocoderName ¥"mygeocoder¥"</pre>
地理情報列の登録抹消	<p>db2se unregister_spatial_column</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_unregister_spatial_column ストアード・プロシージャーのものと同じです。</p> <p>次の例では、表 MYTABLE の MYCOLUMN という名前の地理情報列の登録を抹消します。</p> <pre>db2se unregister_spatial_column mydb -tableName ¥"mytable¥" -columnName ¥"mycolumn¥"</pre>
座標システム定義の更新	<p>db2se alter_cs</p> <p>コマンド固有のパラメーターと必要な権限は、db2gse.ST_alter_coordsysdb2gse.ST_alter_coordsys ストアード・プロシージャーのものと同じです。</p> <p>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。</p> <pre>db2se alter_cs mydb -coordsysName ¥"mycoordsys¥" -organization myNeworganizationb -tableName ¥"mytable¥"</pre>

表 8. 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	<p>db2se alter_srs</p> <p>コマンド固有のパラメーターと必要な権限は、 db2gse.ST_alter_srsdb2gse.ST_alter_srs ストアド・プロシージャのものと同じです。</p> <p>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。</p> <pre>db2se alter_srs mydb -srsName ¥"mysrs¥" -x0ffset 35 -description "This is my own spatial reference system."</pre>

コマンド

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 用のアプリケーションを作成する

DB2 Spatial Extender のストアード・プロシージャや関数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、以下のタスクと参照情報を参照してください。

関連概念:

- 143 ページの『DB2 Spatial Extender サンプル・プログラム』

関連タスク:

- 142 ページの『アプリケーションから DB2 Spatial Extender のストアード・プロシージャを呼び出す』
- 141 ページの『DB2 Spatial Extender のヘッダー・ファイルを地理情報アプリケーションに組み込む』

DB2 Spatial Extender のヘッダー・ファイルを地理情報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・プロシージャや関数を呼び出す場合は、地理情報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。

手順:

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するには、以下のようにします。

1. 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` ディレクトリにあります。`db2path` は、DB2 Universal Database がインストールされているインストール・ディレクトリです。

2. コンパイル・オプションを付けて、`makefile` に `include` ディレクトリのパスが指定されていることを確認する。

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

| 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/spatial/makefile.nt ファイル中の 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に修正します。修正は以下のように行います。

```
| DB2_LIBS = $(DB2_DIR)\lib¥Win64¥db2cli.lib $(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・プロシージャを呼び出す

地理情報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・プロシージャが作成されます。DB2 Spatial Extender のストアード・プロシージャ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・プロシージャの名前を指定します。

手順:

DB2 Spatial Extender のストアード・プロシージャを呼び出すには、以下のようになります。

- 地理情報操作用にデータベースを使用可能にする ST_enable_db ストアード・プロシージャを呼び出すには、次のようにストアード・プロシージャ名を指定します。

```
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・プロシージャです (db2gse! がこの例です)。

- その他の DB2 Spatial Extender のストアード・プロシージャを呼び出すには、次のフォーマットでストアード・プロシージャ名を指定します。db2gse は、すべての DB2 Spatial Extender ストアード・プロシージャのスキーマ名であり、*spatial_procedure_name* はストアード・プロシージャの名前です。

```
CALL db2gse.spatial_procedure_name
```

上の呼び出しには、感嘆符は含まれません。

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

表 9.

ストアード・プロシージャ	説明
GSE_export_sde	地理情報列とその関連表を SDE 転送ファイルにエクスポートする。
GSE_import_sde	SDE 転送ファイルをデータベースにインポートする。
ST_alter_coordsys	データベースの座標系の属性を更新する。
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系を作成する。

表 9. (続き)

ストアド・プロシージャ	説明
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	データベース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系を削除する。
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。
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	地理情報列を登録し、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ため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	地理情報列の登録抹消する。

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。Spatial Extender インストールの検査について詳しくは、このトピックの終わりにある『関連タスク』を参照してください。

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

- 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/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\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、データベースを地理情報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地理情報分析が行えます。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。

- 地理情報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・プログラムから地理情報ストアード・プロシージャ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。たとえば、DB2 Spatial Extender ストアード・プロシージャ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。たとえば、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・プロシージャに慣れていきます。各ステップで使用されるストアード・プロシージャについて詳しくは、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。

表 10. 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	<ul style="list-style-type: none"> <li data-bbox="760 264 1458 495"> • 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。地理情報操作用に使用可能にされているデータベースには、地理情報タイプのセット、空間処理関数セット、地理情報述部セット、新規の索引タイプと 1 組の地理情報のカタログ表とビューが入っています。 <li data-bbox="760 506 1458 894"> • 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地理情報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地理情報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、地理情報タイプのセット、空間処理関数セット、地理情報述部セット、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地理情報カタログ表とビューを除去します。 <p data-bbox="760 810 1458 894">• 空間データベースを使用可能にする 上記に同じです。</p>
座標システムを作成またはドロップする	<ul style="list-style-type: none"> <li data-bbox="760 926 1458 1052"> • 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。 <li data-bbox="760 1062 1458 1220"> • 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。 <li data-bbox="760 1230 1458 1377"> • 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	<ul style="list-style-type: none"> <li data-bbox="760 1409 1458 1682"> • SRSDEMO1 という名前の地理情報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、地理情報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、図形データが収められます。SRS が特定の地理情報列に対して登録されると、その地理情報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。 <li data-bbox="760 1692 1458 1850"> • 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。 <li data-bbox="760 1860 1458 1892"> • KY_STATE_SRS という名前の SRS を作成する

表 10. 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
地理情報表を作成し、データを入れる	<ul style="list-style-type: none"> • CUSTOMERS 表を作成する • CUSTOMERS 表にデータを追加する <p>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。</p> <ul style="list-style-type: none"> • LOCATION 列を追加して CUSTOMER 表を変更する <p>ALTER TABLE ステートメントによってタイプが ST_Point 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。</p> <ul style="list-style-type: none"> • OFFICES 表を作成する <p>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。</p>
列にデータを入れる	<ul style="list-style-type: none"> • 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・データをジオコードする <p>このステップでは、ジオコーダー・ユーティリティを呼び出して、地理情報のバッチ・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ドする、または再ジオコードする必要があるときに行われます。</p> <ul style="list-style-type: none"> • 地理情報参照システム 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 <p>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 ユーティリティは、既存の表に新しいレコードを追加します。</p> <ul style="list-style-type: none"> • 地理情報参照システム 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 <p>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 ユーティリティは、ロードする前に表を作成します。</p> <ul style="list-style-type: none"> • 地理情報参照システム 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

表 10. 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
ジオコーダーを登録または登録抹消する	<ul style="list-style-type: none"> • 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を登録抹消する • ジオコーダー KY_STATE_GC を登録する <p>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。</p>
地理情報索引を作成する	<ul style="list-style-type: none"> • CUSTOMERS 表の LOCATION 列の地理情報格子索引を作成する • CUSTOMERS 表の LOCATION 列の地理情報格子索引をドロップする • CUSTOMERS 表の LOCATION 列の地理情報格子索引を作成する • OFFICES 表の LOCATION 列の地理情報格子索引を作成する • FLOODZONES 表の LOCATION 列の地理情報格子索引を作成する • REGIONS 表の LOCATION 列の地理情報格子索引を作成する <p>以上のステップでは、CUSTOMERS、OFFICES、FLOODZONES および REGIONS 表の地理情報格子索引を作成します。</p>
自動ジオコーディングを使用可能にする	<ul style="list-style-type: none"> • 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 <p>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。</p> <ul style="list-style-type: none"> • 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 <p>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。</p>
CUSTOMERS 表に対する挿入、更新、削除操作を実行する	<ul style="list-style-type: none"> • 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 <p>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 列にもジオコードされます。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。</p>

表 10. 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不可にする	<ul style="list-style-type: none"> • 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 列の地理情報索引をドロップする <p>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地理情報索引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。</p> <p>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地理情報索引をドロップし、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。</p>
CUSTOMERS 表を再ジオコードする	<ul style="list-style-type: none"> • より低いレベルの精度 (100% ではなく 90%) で、CUSTOMERS 表の LOCATION 列を再ジオコードする • CUSTOMERS 表の LOCATION 列の地理情報索引を再作成する • 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 <p>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、地理情報索引を再作成し、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。地理情報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ドした後、自動ジオコーディングがもう一度使用可能にされ、地理情報索引が再作成されます。これによって、これ以降の挿入と更新操作のために、地理情報索引と地理情報列の増分保守が可能になります。</p>
ビューを作成し、ビューに地理情報列を登録する	<ul style="list-style-type: none"> • 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 • ビューの地理情報列を登録する <p>このステップでは、ビューを作成し、その地理情報列を登録します。</p>

表 10. 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
地理情報を分析する	<ul style="list-style-type: none"> • 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ゾーン数を調べる (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) • 特定の洪水ゾーン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) <p>以上のすべてのステップで、sqlRunSpatialQueries ストアド・プロシージャが使用されます。</p> <p>これらのステップは、DB2 SQL の関数と地理情報述部を使用して、地理情報を分析します。DB2 照会オプティマイザーは、空間列に関する地理情報索引を活用して、可能なかぎり、照会パフォーマンスを向上させます。</p>
空間データを形状ファイルにエクスポートする	<ul style="list-style-type: none"> • HIGHRISKCUSTOMERS ビューを形状ファイルにエクスポートする <p>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。</p> <p>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。</p>
SDE ファイルをエクスポートおよびインポートする	<ul style="list-style-type: none"> • CUSTOMERS 表を SDE 転送ファイルにエクスポートする • 新しくエクスポートした SDE 転送ファイルからデータをインポートする <p>これらのステップは、SDE 転送ファイルのエクスポートとインポートの例を示しています。</p> <p>これらのステップは、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 SDE 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。</p>

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

関連タスク:

- 41 ページの『Spatial Extender のインストールの検査』
- 43 ページの『インストールの問題のトラブルシューティング』
- 141 ページの『DB2 Spatial Extender 用のアプリケーションを作成する』
- 142 ページの『アプリケーションから DB2 Spatial Extender のストアード・プロシージャを呼び出す』
- 141 ページの『DB2 Spatial Extender のヘッダー・ファイルを地理情報アプリケーションに組み込む』

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・プロシージャおよび関数を使用すると、DB2 はストアード・プロシージャ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。

この章では、それらのアプローチをそれぞれ説明しています。

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
- DB2 Spatial Extender 関数
- DB2 Spatial Extender コマンド行プロセッサ (CLP)
- DB2 コントロール・センター

要求した地理情報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: The operation was completed successfully.
```

表 11. 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。

表 11. 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
The operation was completed successfully.	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。たとえば、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・コードを GSE4107N と想定した場合、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 Grid size value "<grid-size>" is not valid where it is used.
```

```
Explanation: The specified grid size "<grid-size>" is not valid.
```

```
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).

User Response: Specify a valid value for the grid size.

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。

関連概念:

- 153 ページの『DB2 Spatial Extender ストアード・プロシージャー出力パラメーター』
- 156 ページの『DB2 Spatial Extender 関数メッセージ』
- 157 ページの『DB2 Spatial Extender CLP メッセージ』
- 159 ページの『DB2 コントロール・センター・メッセージ』
- 162 ページの『管理通知ファイル』

関連タスク:

- 160 ページの『db2trc コマンドによる DB2 Spatial Extender の問題のトレース』

関連資料:

- 「メッセージ・リファレンス 第 1 巻」の『GSE メッセージ』

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 を使用可能にして使用するとき、または DB2 Spatial Extender CLP (db2se) を使用するとき、**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。たとえば、

- 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・プロシージャの msg_code 番号は、次の表
に示すように 3 つのカテゴリに分けられます。

表 12. ストアード・プロシージャ・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
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。ユーザーは
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 ストアド・プロシージャ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、地理情報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、ストアド・プロシージャの実行後に表示されます。

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

```
Value of output parameters
```

```
-----  
Parameter Name : MSGCODE  
Parameter Value : 0
```

```
Parameter Name : MSGTEXT  
Parameter Value : GSE0000I The operation was completed successfully.
```

```
Return Status = 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。たとえば、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  
         geometry 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
```

関連概念:

- 151 ページの『DB2 Spatial Extender メッセージの解釈方法』
- 156 ページの『DB2 Spatial Extender 関数メッセージ』
- 157 ページの『DB2 Spatial Extender CLP メッセージ』
- 159 ページの『DB2 コントロール・センター・メッセージ』

関連資料:

- 「メッセージ・リファレンス 第 1 巻」の『GSE メッセージ』

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。たとえば、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 13. 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。
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。
Polygon is not closed	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。 <ul style="list-style-type: none"> • Spatial Extender 関数エラーに対する SQLSTATE コードは 38Sxx 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 01HSx 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 Polygon is not closed 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。

関連概念:

- 151 ページの『DB2 Spatial Extender メッセージの解釈方法』
- 153 ページの『DB2 Spatial Extender ストアード・プロシージャ出力パラメーター』
- 157 ページの『DB2 Spatial Extender CLP メッセージ』
- 159 ページの『DB2 コントロール・センター・メッセージ』

関連資料:

- 「メッセージ・リファレンス 第 1 巻」の『GSE メッセージ』

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 (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・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・プロシージャを呼び出すと、ストアード・プロシージャ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。たとえば、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、たとえば、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 The operation was completed successfully.
```

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ド・プロシージャを呼び出す場合、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 によって戻されるマイグレーション・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。

コマンド `db2se migrate mydb -messagesFile /tmp/migrate.msg` を使用して、データベース `mydb` のマイグレーションを呼び出すものと想定します。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。

```
Migrating database. Please wait ...
GSE0000I The operation was completed successfully.
```

関連概念:

- 151 ページの『DB2 Spatial Extender メッセージの解釈方法』
- 153 ページの『DB2 Spatial Extender ストアード・プロシージャ出力パラメータ』
- 156 ページの『DB2 Spatial Extender 関数メッセージ』
- 159 ページの『DB2 コントロール・センター・メッセージ』

関連資料:

- 「メッセージ・リファレンス 第 1 巻」の『GSE メッセージ』

DB2 コントロール・センター・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 (DB2 Message)」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。

DB2 Spatial Extender メッセージ:

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 (DB2 Message)」ウィンドウのテキスト域に表示されます。たとえば、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 (DB2 Message)」ウィンドウの左側に表示されます。たとえば 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 (DB2 Message)」ウィンドウのテキスト域に表示されます。

「DB2 メッセージ (DB2 Message)」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ます、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>" is a duplicate name. 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.

関連概念:

- 151 ページの『DB2 Spatial Extender メッセージの解釈方法』
- 153 ページの『DB2 Spatial Extender ストアード・プロシージャ出力パラメータ』
- 156 ページの『DB2 Spatial Extender 関数メッセージ』
- 157 ページの『DB2 Spatial Extender CLP メッセージ』

関連資料:

- 「メッセージ・リファレンス 第 1 巻」の『GSE メッセージ』

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 SYSMOINT 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。

手順:

DB2 Spatial Extender イベントをメモリーにトレースするには、以下の基本ステップに従います。

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. トレースをファイルにダンプします。たとえば、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。たとえば、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
```

問題が発生したら直ちにトレースをダンプします。

5. トレースをオフにします。たとえば、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。たとえば、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。たとえば、以下のように指定します。

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

関連概念:

- 「トラブルシューティング・ガイド」の『DB2 トレース (db2trc)』
- 151 ページの『DB2 Spatial Extender メッセージの解釈方法』
- 162 ページの『管理通知ファイル』

関連資料:

- 「メッセージ・リファレンス 第 1 巻」の『GSE メッセージ』

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・マネージャ構成パラメーターで指定されたディレクトリにあります。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 (たとえば、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。

関連概念:

- 「トラブルシューティング・ガイド」の『管理ログの解釈』
- 151 ページの『DB2 Spatial Extender メッセージの解釈方法』

関連タスク:

- 160 ページの『db2trc コマンドによる DB2 Spatial Extender の問題のトレース』

関連資料:

- 「メッセージ・リファレンス 第 1 巻」の『GSE メッセージ』

プロジェクトの作成

第 4 部 DB2 Geodetic Extender の使用

第 16 章 DB2 Geodetic Extender

この章では、DB2 Geodetic Extender 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 Extender

DB2[®] Geodetic Extender では、地球を球体として扱うことができます。Geodetic Extender では、他の Spatial Extender 機能と同じデータ・タイプ、関数を使用して、極の周囲のデータおよび 180 度の子午線を超え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 Extender 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 Extender は、地球表面上に存在する物体を高い精度で取り扱えるよう設計されています。

そのために、Geodetic Extender では、平面の *xy* 座標系ではなく、楕円体地球モデルまたは測地原点に基づく緯度経度座標系を使用しています。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。詳細については、169 ページの『測地緯度と測地経度』、62 ページの『測地座標系』、67 ページの『投影座標系』を参照してください。

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。この座標系には、2000000000 から 2000001000 までの範囲の空間参照系 ID (SRID) を持たせます。Geodetic Extender は、318 の定義済み測地空間参照系を提供しています。

DB2 Geodetic Extender 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。DB2 Spatial Extender とは別に DB2 Geodetic Extender のみを注文することもできますが、その場合は Geodetic Extender のみのライセンスを別に購入してください。

関連概念:

- 168 ページの『DB2 Geodetic Extender を使用すべき場合と DB2 Spatial Extender を使用すべき場合』
- 168 ページの『測地基準』

関連タスク:

- 175 ページの『DB2 Geodetic Extender のセットアップと使用可能化』

関連資料:

- 223 ページの『DB2 Geodetic Extender によってサポートされる測地系』

DB2 Geodetic Extender を使用すべき場合と DB2 Spatial Extender を使用すべき場合

DB2[®] Spatial Extender と DB2 Geodetic Extender はいずれも、DB2 データベースの空間システム (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 Extender 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系を使用します。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。Geodetic Extender の基礎となっているのは、Geodyssey Limited からライセンスを取得した 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。

Geodetic Extender は、地球上の (一度の地図投影では十分な正確さが得られないほどの) 広い領域 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適しています。

- 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、どのような方法を探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 (投影座標系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。たとえば、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。

関連概念:

- 167 ページの『DB2 Geodetic Extender』
- 172 ページの『測地領域』
- 169 ページの『測地緯度と測地経度』
- 171 ページの『測地距離』
- 232 ページの『測地回転楕円体』

関連タスク:

- 175 ページの『DB2 Geodetic Extenderのセットアップと使用可能化』

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 ディメンション図形です。回転楕円体の詳細については、62 ページの『測地座標系』を参照してください。

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 Extender のサポートしている測地系であれば、どれでも選択できます。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。

- 223 ページの『DB2 Geodetic Extender によってサポートされる測地系』には、Geodetic Extender の提供している 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。詳しくは、77 ページの『空間参照系の作成』を参照してください。

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 Extender は、測地系変換を行いません。

関連概念:

- 168 ページの『DB2 Geodetic Extender を使用すべき場合と DB2 Spatial Extender を使用すべき場合』
- 172 ページの『測地領域』
- 169 ページの『測地緯度と測地経度』
- 171 ページの『測地距離』
- 232 ページの『測地回転楕円体』

関連タスク:

- 71 ページの『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』
- 77 ページの『空間参照系の作成』

関連資料:

- 223 ページの『DB2 Geodetic Extender によってサポートされる測地系』

測地緯度と測地経度

DB2 Geodetic Extender 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。

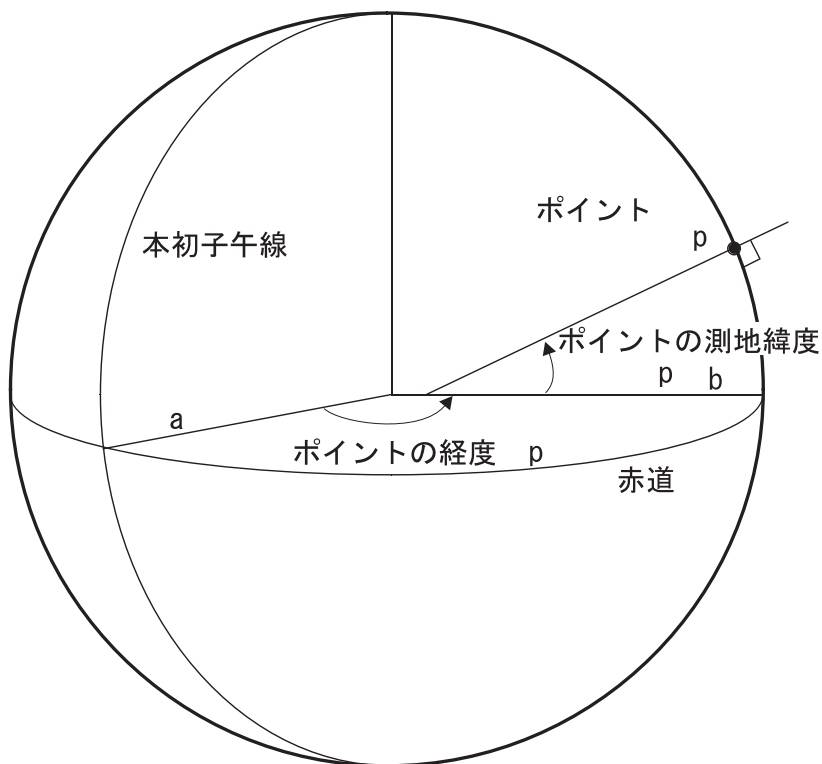


図 17. 測地緯度と測地経度に対応する角度

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° 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。

関連概念:

- 167 ページの『DB2 Geodetic Extender』
- 168 ページの『DB2 Geodetic Extender を使用すべき場合と DB2 Spatial Extender を使用すべき場合』
- 172 ページの『測地領域』
- 168 ページの『測地基準』
- 171 ページの『測地距離』
- 232 ページの『測地回転楕円体』

測地距離

DB2[®] Geodetic Extender 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。

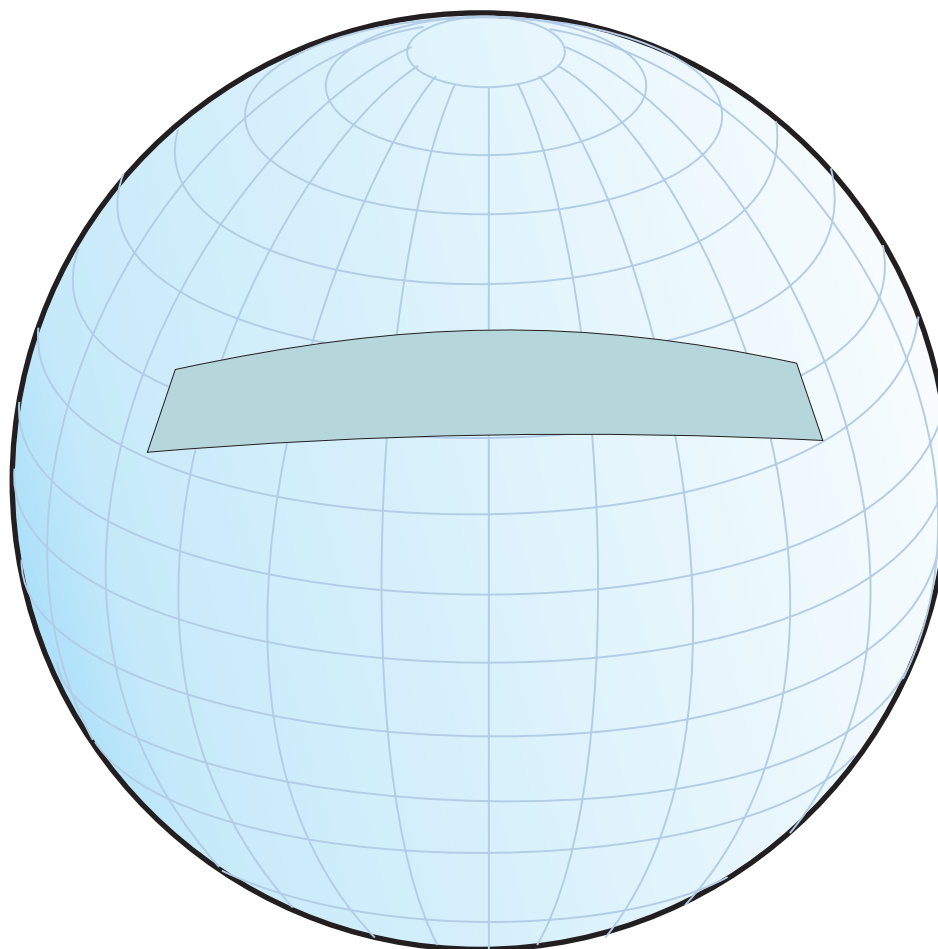


図 18. ポイント間の距離が長いポリゴンの囲む領域

測地線とは異なる線、たとえば、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。

関連概念:

- 62 ページの『測地座標系』

関連資料:

- 209 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』
- 383 ページの『ST_Distance』

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 Extender は、閉じ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。

172 ページの図 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。

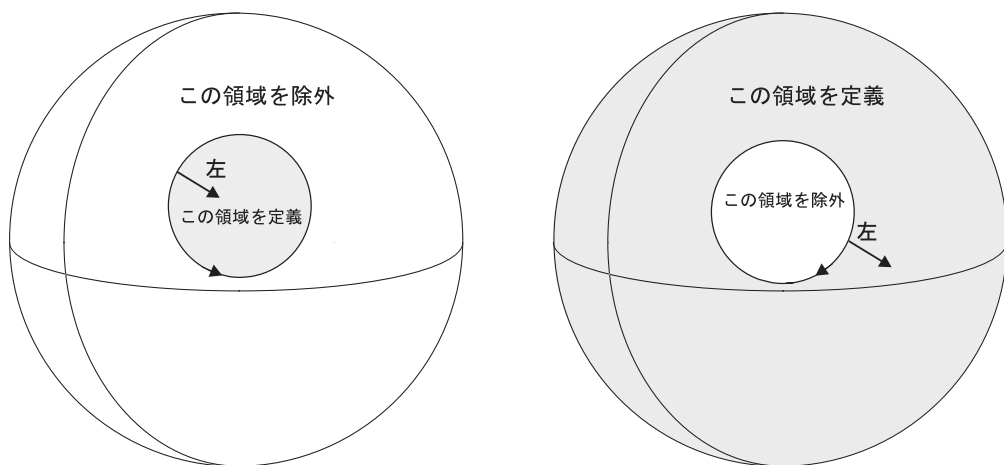


図 19. 領域の定義と除外

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、ポリゴン内部のポイントが、指定した頂点の左側に位置することになる、ということ considering 頂点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、173 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。173 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。

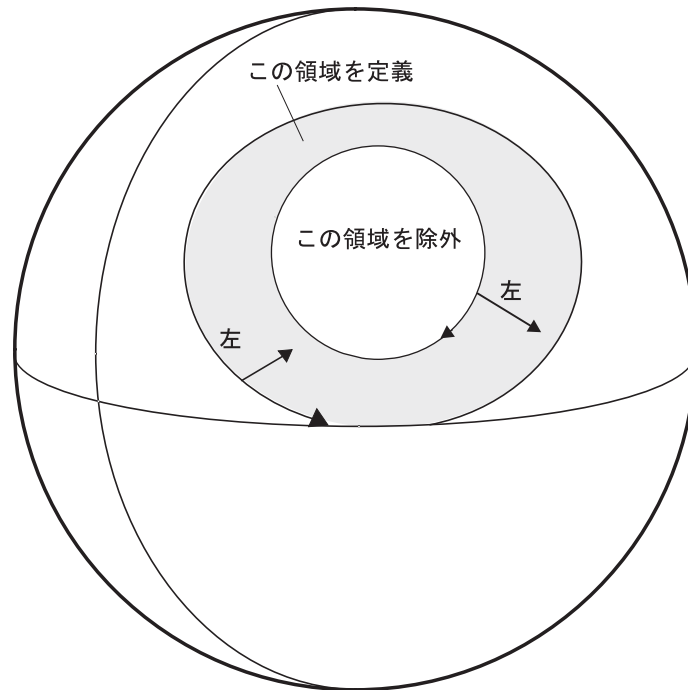


図 20. 複数のリングでの領域の定義

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、このメッセージが送信
 されます。

```
GSE3733W "Polygon covers more than half the earth. Verify counter-clockwise
orientation of the vertex points."
```

関連概念:

- 67 ページの『投影座標系』
- 168 ページの『測地基準』
- 70 ページの『空間参照系』

関連タスク:

- 71 ページの『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』
- 77 ページの『空間参照系の作成』

第 17 章 DB2 Geodetic Extender のセットアップ

この章では、DB2 Geodetic Extender 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 Extenderのセットアップと使用可能化

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 Extender では地球を球体として扱います。Geodetic Extender をインストールすると、空間データを平面地図を使った場合よりも正確に分析できるようになります。

DB2 Geodetic Extender システムは、DB2 Universal Database、DB2 Spatial Extender、DB2 Geodetic Extender から構成され、ほとんどのアプリケーションでは、空間ブラウザーも加わります。

推奨事： DB2 Geodetic Extender の改良、機能追加などについての情報を得るには、「DB2 リリース・ノート」を参照してください。

前提条件:

DB2 Geodetic Extender を使用可能化する前には、以下のことを行う必要があります。

- DB2 Universal Database™ Enterprise Server Edition バージョン 8.2 のインストールと構成。

DB2 Spatial Extender と DB2 Geodetic Extender をインストールする前には DB2 UDB をシステムに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 Universal Database 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 Extender 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 Extender が含まれています。Spatial Extender ディスク・スペース所要量には、Geodetic Extender に必要な分も含まれています。ただし、Geodetic Extender のライセンスを購入し、Geodetic Extender を使用可能化するまでは、Geodetic Extender を使用することはできません。詳細については、26 ページの『Spatial Extender をインストールするためのシステム要件』と 25 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。

- DB2 Spatial Extender バージョン 8.1 データベースを持っている場合は、DB2 Geodetic Extender を使用する前にマイグレーションする必要がある。

Geodetic Extender は、測地データを扱うため、いくつかの空間処理関数を再定義し、Spatial Extender とは異なる測地参照系を定義します。マイグレーション・ユ

DB2 Geodetic Extender のセットアップ

ーティリティー **migrate_v82** を使用すれば、既存の空間データ対応のデータベースを、測地データを扱えるものにできます。詳細については、47 ページの『地理情報操作が可能になっているデータベースのマイグレーション』を参照してください。

- DB2 Geodetic Extender のライセンスの購入

DB2 Geodetic Extender ライセンスを購入する際には、Geodetic ライセンス・キーを使用可能化できます。DB2 Geodetic Extender を購入する場合は、営業担当者に連絡をしてください。

制約事項:

DB2 Geodetic Extender は、DB2 Universal Database™ Enterprise Server Edition バージョン 8.2 にのみライセンスされます。

手順:

DB2 Geodetic Extender 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 (License Center)」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。

DB2 Geodetic Extender ライセンスを使用可能化したら、183 ページの『空間列に測地データを入れる』。

関連概念:

- 26 ページの『Spatial Extender をインストールするためのシステム要件』

関連タスク:

- 25 ページの『Spatial Extender のセットアップとインストール』
- 47 ページの『地理情報操作が可能になっているデータベースのマイグレーション』
- 183 ページの『空間列に測地データを入れる』

関連資料:

- 45 ページの『DB2 Spatial Extender のデータと地図の入った CD』

Informix Geodetic DataBlade から DB2 Geodetic Extender へのマイグレーション

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 Extender 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。

前提条件:

Geodetic DataBlade アプリケーションを、DB2 Geodetic Extender 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、DB2 Geodetic Extender へのマイグレーションを可能にするには、以下の条件を満たす必要があります。

- 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 Extender 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。

手順:

IBM Informix Geodetic DataBlade から IBM DB2 Geodetic Extender へのマイグレーションは以下の手順で行います。

1. SQL ステートメントを、DB2 Geodetic Extender のデータ・タイプや関数を利用するよう書き直す。データ・タイプ、関数の対応関係については以下の表を参照。
 - 表 14
 - 178 ページの表 15
 - 179 ページの表 16
 - 179 ページの表 17
 - 180 ページの表 18
 - 181 ページの表 19
 - 181 ページの表 20
 - 181 ページの表 21
 - 181 ページの表 22
 - 182 ページの表 23
2. データを DB2 Geodetic Extender にロードあるいはインポートする。
3. Informix ODBC、ESQL/C、JDBC を利用するアプリケーションを書き直す。183 ページの表 24 に、Geodetic DataBlade と Geodetic Extender 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 14. Informix Geodetic DataBlade と Geodetic Extender のデータ・タイプの対応関係

Informix Geodetic DataBlade のデータ・タイプ	DB2 Geodetic Extender 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
GeoBox		まず Geodetic DataBlade で GeoPolygon に変換し、Geodetic Extender で ST_Polygon を使用する。
GeoCircle		まず GeoPolygon に変換した後、ST_Polygon にマイグレーションする。
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	

DB2 Geodetic Extender のセットアップ

表 14. Informix Geodetic DataBlade と Geodetic Extender のデータ・タイプの対応関係 (続き)

Informix Geodetic DataBlade のデータ・タイプ	DB2 Geodetic Extender 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
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 Extender のデータ・タイプはありません。

- GeoAltitude
- GeoAltRange
- GeoAngle
- GeoAzimuth
- GeoBox
- GeoCircle
- GeoCoords
- GeoDistance
- GeoEllipse
- GeoLatitude
- GeoLongitude
- GeoTimeRange
- GeoVoronoi

表 15. Informix Geodetic DataBlade と Geodetic Extender の述部関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Extender の対応関数
Contains	ST_Contains
Inside	ST_Within
Intersect	ST_Intersects
Outside	ST_Disjoint
Within	ST_Distance

以下の Geodetic DataBlade 述部関数には、対応する Geodetic Extender の関数はありません。

- Beyond
- Equal
- Nearest

表 16. Informix Geodetic DataBlade と Geodetic Extender のプロダクション関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Extender 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 17. Informix Geodetic DataBlade と Geodetic Extender のアクセサー関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Extender 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。
Coords	ST_PointN	
Dimension	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。	

DB2 Geodetic Extender のセットアップ

表 17. Informix Geodetic DataBlade と Geodetic Extender のアクセサー関数の対応関係 (続き)

Informix Geodetic DataBlade の関数	DB2 Geodetic Extender の対応関数	類似の関数に関するコメント
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 を使用する。
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	

以下の Geodetic DataBlade アクセサー関数には、対応する Geodetic Extender の関数はありません。

- IsLarge
- IsSmallArea

表 18. Informix Geodetic DataBlade と Geodetic Extender の修飾関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Extender の対応関数
SetSRID	ST_SRID

以下の Geodetic DataBlade 修飾関数には、対応する Geodetic Extender の関数はありません。

- SetAltRange
- SetAltRangeZ
- SetDist
- SetTimeRange

表 19. Informix Geodetic DataBlade と Geodetic Extender のメジャー関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Extender の対応関数
Area	ST_Area
Distance	ST_Distance
長さ	ST_Length、ST_Perimeter

VoronoiResolution メジャー関数には、対応する Geodetic Extender の関数はありません。

表 20. Informix Geodetic DataBlade と Geodetic Extender のダウン cast 関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Extender 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 21. Informix Geodetic DataBlade と Geodetic Extender のコンストラクター関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Extender の対応関数
GeoCoords	ST_Point
GeoPoint	ST_Point

以下の Geodetic DataBlade コンストラクター関数には、対応する Geodetic Extender の関数はありません。

- GeoBox
- GeoCircle
- GeoEllipse
- GeoLineseg

表 22. Informix Geodetic DataBlade と Geodetic Extender の診断関数の対応関係

Informix Geodetic DataBlade の関数	DB2 Geodetic Extender の対応関数
GeoTraceLevel	DB2 Trace Facility
IsValidGeometry	ST_IsValid

以下の Geodetic DataBlade 診断関数には、対応する Geodetic Extender の関数はありません。

- GeoInRowSize
- GeoOutOfRowSize
- GeoRelease
- GeoTotalSize

DB2 Geodetic Extender のセットアップ

- GeoTraceLevelSet
- GeoWarningLevel
- GeoWarningLevelSet
- IsValidSDTS

表 23. Informix Geodetic DataBlade と Geodetic Extender 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 Extender の対応カタログ・ビュー
GeoLenUnit	DB2GSE.ST_UNITS_OF_MEASURE
GeoSpatialRef	DB2GSE.SPATIAL_REF_SYS

以下の Geodetic DataBlade システム・カタログ表には、対応する Geodetic Extender の表やビューはありません。

- GeoEllipsoid
- GeoParam
- GeoVoronoi

以下の Geodetic DataBlade のユーザー設定可能なパラメーター関数には、対応する Geodetic Extender の関数はありません。

- GeoParamSessionGet
- GeoParamSessionSet

以下の Geodetic DataBlade の AltRange 関数には、対応する Geodetic Extender の関数はありません。

- AltRange
- Bottom
- Contains
- Equal
- Inside
- Intersect
- IsAny
- Outside
- Top

以下の Geodetic DataBlade の TimeRange 関数には、対応する Geodetic Extender の関数はありません。

- Begin
- Contains
- End
- Equal
- IsAny
- Inside
- Intersect

- Outside
- TimeRange

以下の Geodetic DataBlade 楕円関数には、対応する Geodetic Extender の関数はありません。

- Azimuth
- Coords
- Major
- Minor

以下の Geodetic DataBlade 円関数には、対応する Geodetic Extender の関数はありません。

- Coords
- Radius

以下の Geodetic DataBlade 角度演算関数には、対応する Geodetic Extender の関数はありません。

- Divide
- Minus
- Negate
- Plus
- Times

表 24. Geodetic DataBlade と DB2 Geodetic Extender 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 Extender の 対応クライアント接続
ESQLC	SQC
ODBC	ODBC
JDBC	JDBC

以下の Geodetic DataBlade のクライアント接続には、対応する Geodetic Extender のクライアント接続はありません。

- Java API
- LIBMI

空間列に測地データを入れる

空間列を作成し、空間索引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。測地データは以下の方法で入れることができます。

- 以下の形式のデータを新規、または既存の表にインポートする。
 - 形状
 - SDE
- 以下の形式のデータの値を挿入、あるいは更新する。
 - 形状
 - SDE

DB2 Geodetic Extender のセットアップ

- 事前割り当てテキスト (WKT)
- 事前割り当てバイナリー (WKB)
- GML (地理マークアップ言語)

制約事項:

- Spatial Extender Version 8.2 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・プロシージャは使用できません。
- 測地データに対応するためには、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。詳しくは、70 ページの『空間参照系』を参照してください。
- 形状データ、SDE 転送データは、地理座標系で表現されている必要があります。詳しくは、62 ページの『測地座標系』を参照してください。

手順:

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。詳細については、91 ページの『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』および 92 ページの『SDE 転送データを新規の表または既存の表にインポートする』を参照してください。

関連概念:

- 70 ページの『空間参照系』
- 62 ページの『測地座標系』

関連タスク:

- 91 ページの『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』
- 92 ページの『SDE 転送データを新規の表または既存の表にインポートする』
- 87 ページの『空間列の登録』

第 18 章 測地索引

測地ポロノイ・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。

- 測地ポロノイ・インデックスについての説明
- ポロノイ・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
- 測地ポロノイ・インデックスの作成方法についての説明

測地ポロノイ・インデックス

DB2[®] Geodetic Extender は、測地データへのアクセスを高速化するのに役立つ測地ポロノイ・インデックスを提供しています。この索引は、地球表面のポ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。詳細については、186 ページの『ポロノイ・セル構造』を参照してください。

Geodetic Extender は、図形ごとに最小外接円 (MBC) を算出します。MBC とは、測地図形を囲む円のことで、ポロノイ・インデックスでは、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。ポロノイ・インデックス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。ポロノイ・インデックスを利用すれば、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。ポロノイ・インデックス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 文節中に以下の関数を含む照会のすべてについて測地ポロノイ・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

詳細については、128 ページの『照会を最適化するために索引を使用する関数』を参照してください。

測地ポロノイ・インデックスを作成する際には、代替ポロノイ・セル構造を選択することもできます。詳細については、187 ページの『代替ポロノイ・セル構造を選択する際の考慮事項』を参照してください。

関連概念:

- 186 ページの『ボロノイ・セル構造』
- 187 ページの『代替ボロノイ・セル構造を選択する際の考慮事項』
- 104 ページの『空間グリッド・インデックス』

関連タスク:

- 189 ページの『測地ボロノイ・インデックスの作成』

関連資料:

- 192 ページの『DB2 Geodetic Extender に提供されているボロノイ・セル構造』
- 128 ページの『照会を最適化するために索引を使用する関数』

ボロノイ・セル構造

演算を効率よく行うため、DB2[®] Geodetic Extender は、地球表面を小さく扱いやすい、蜂の巣のようなセルに分割します。この分割は、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、他のどの格子点よりも特定の格子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。

たとえば、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。

187 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 Extender はこのセル構造を、空間演算に利用します。

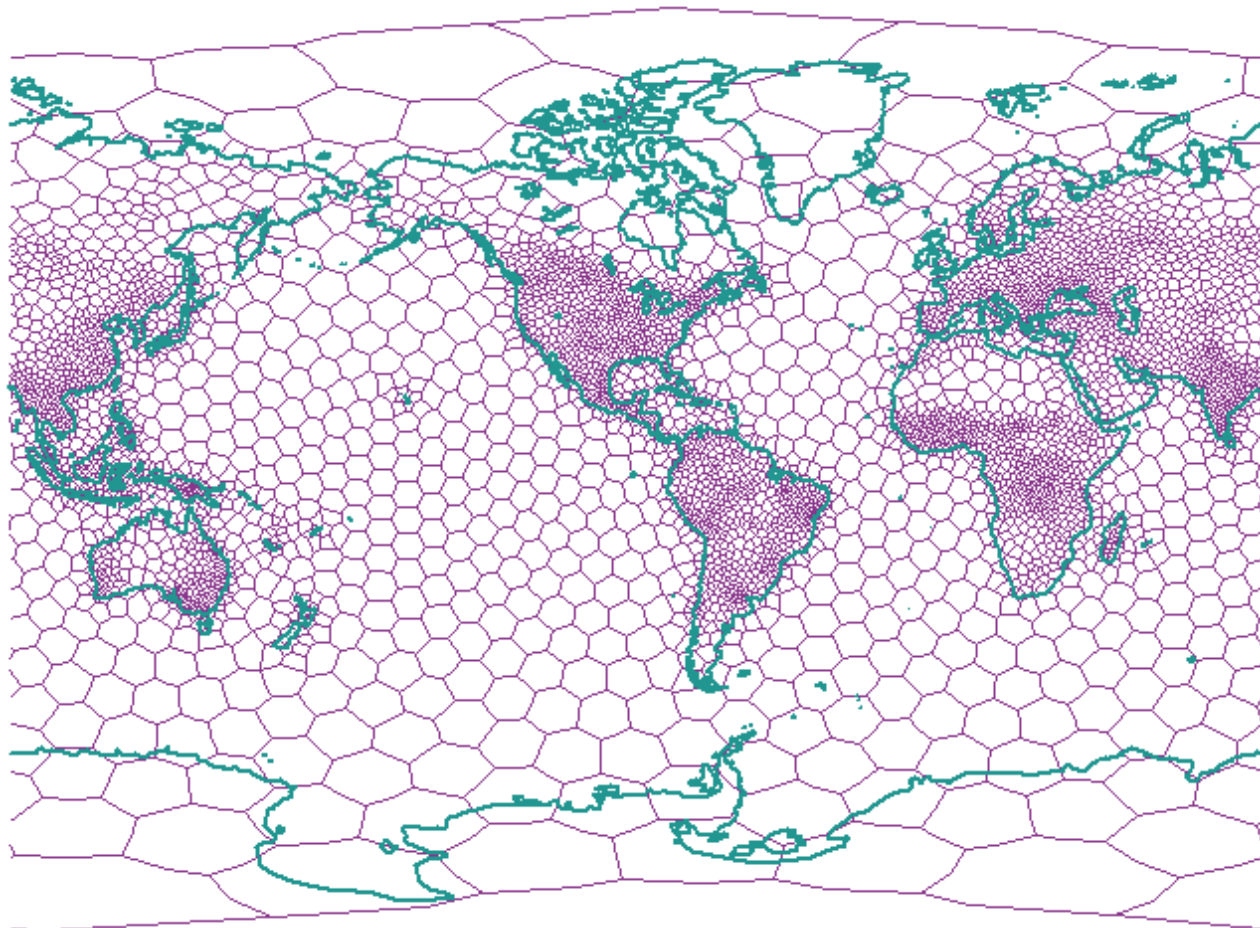


図 21. 世界の人口密度を基準にしたボロノイ構造の例

関連概念:

- 185 ページの『測地ボロノイ・インデックス』
- 187 ページの『代替ボロノイ・セル構造を選択する際の考慮事項』

関連タスク:

- 189 ページの『測地ボロノイ・インデックスの作成』

関連資料:

- 190 ページの『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』
- 192 ページの『DB2 Geodetic Extender に提供されているボロノイ・セル構造』

代替ボロノイ・セル構造を選択する際の考慮事項

測地図形に対する操作では、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 Extender

は、個々のユーザーが自らのデータに適合するものを選べるよう、多数の ボロノイ・セル構造を提供しています。利用できる代替セル構造のリスト、およびそれぞれのセルが構造がどのようなものかを示す図は、192 ページの『DB2 Geodetic Extender に提供されているボロノイ・セル構造』に記載されています。

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 以下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

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・セル構造を変更することで、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 (たとえば、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。

関連概念:

- 185 ページの『測地ボロノイ・インデックス』
- 186 ページの『ボロノイ・セル構造』

関連タスク:

- 189 ページの『測地ボロノイ・インデックスの作成』

関連資料:

- 190 ページの『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』
- 192 ページの『DB2 Geodetic Extender に提供されているボロノイ・セル構造』

測地ボロノイ・インデックスの作成

DB2 Geodetic Extender 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操作手法をサポートしています。索引を使用すれば、照会の実行は速くなります。

前提条件:

測地ボロノイ・インデックスを作成する際には、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 (111 ページの『地理情報格子索引の作成』を参照)。

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 UDB は、処理中に別名を解決します。

手順:

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 (Create Index)」ウィンドウを使用する。
- EXTEND USING 文節で 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。

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方法 :

1. コントロール・センターで、測地ボロノイ・インデックスを作成しようとする空間列が入っている表を右クリックし、ポップアップ・メニューから「**Spatial Extender**」→「**空間インデックス (Spatial Indexes)**」を選択する。「空間インデックス」ウィンドウが開きます。
2. 「空間インデックス」ウィンドウのオンライン・ヘルプの説明に従う。「空間インデックス」ウィンドウの「**ヘルプ (Help)**」ボタンをクリックすると、これらの説明を表示できます。

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方法 :

```
EXTEND USING 文節と db2gse.spatial_index 格子索引拡張子を使用して  
CREATE INDEX コマンドを出します。
```

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)
```

測地ボロノイ・インデックスの場合、USING db2gse.spatial_index 文節の最初の 2 つのパラメーターに値「-1」を指定する必要があります。詳細につ

いては、『測地ポロノイ・インデックスを作成するための CREATE INDEX ステートメント』を参照してください。

関連概念:

- 185 ページの『測地ポロノイ・インデックス』
- 186 ページの『ポロノイ・セル構造』
- 187 ページの『代替ポロノイ・セル構造を選択する際の考慮事項』

関連タスク:

- 111 ページの『地理情報格子索引の作成』

関連資料:

- 190 ページの『測地ポロノイ・インデックスを作成するための CREATE INDEX ステートメント』
- 192 ページの『DB2 Geodetic Extender に提供されているポロノイ・セル構造』
- 128 ページの『照会を最適化するために索引を使用する関数』

測地ポロノイ・インデックスを作成するための CREATE INDEX ステートメント

測地ポロノイ・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 文節を指定して使用してください。

構文:

```
▶ CREATE INDEX index_schema.index_name ON
▶ table_schema.table_name (column_name) EXTEND USING
▶ db2gse.spatial_index (-1, -1, Voronoi_ID)
```

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 UDB 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 UDB 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

測地ポロノイ・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 Extenderによる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・セル構造に対応します。

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 文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・インデックス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・インデックスの使用例です。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 文節で 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)',
20000000000)
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 文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ブ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
```

関連概念:

- 185 ページの『測地ポロノイ・インデックス』
- 186 ページの『ポロノイ・セル構造』
- 187 ページの『代替ポロノイ・セル構造を選択する際の考慮事項』

関連タスク:

- 189 ページの『測地ポロノイ・インデックスの作成』

関連資料:

- 192 ページの『DB2 Geodetic Extender に提供されているポロノイ・セル構造』

DB2 Geodetic Extender に提供されているポロノイ・セル構造

ポロノイ・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、ポロノイ・セル構造の中でも、セルの密度が濃い部分のみを示してあります。ポロノイ・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 Extenderが提供するポロノイ・セル構造のリストです。これらのポロノイ・セル構造は、Geodyssey Ltd. によって提供されているものです。

表 25. ポロノイ・セル構造

説明	ポロノイ ID	対応する図
世界の人口密度を基準にしたもの	1	194 ページの図 22
米国	2	195 ページの図 23
カナダ	3	196 ページの図 24

表 25. ボロノイ・セル構造 (続き)

説明	ボロノイ ID	対応する図
インド	4	197 ページの図 25
日本	5	198 ページの図 26
アフリカ	6	199 ページの図 27
オーストラリア	7	200 ページの図 28
ヨーロッパ	8	201 ページの図 29
北アメリカ	9	202 ページの図 30
南アメリカ	10	203 ページの図 31
地中海	11	204 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	205 ページの図 33
世界の工業生産高を基準にしたもの (G7 諸国)	13	206 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	207 ページの図 35

世界の人口密度を基準にしたセル構造 (ポロノイ ID: 1)

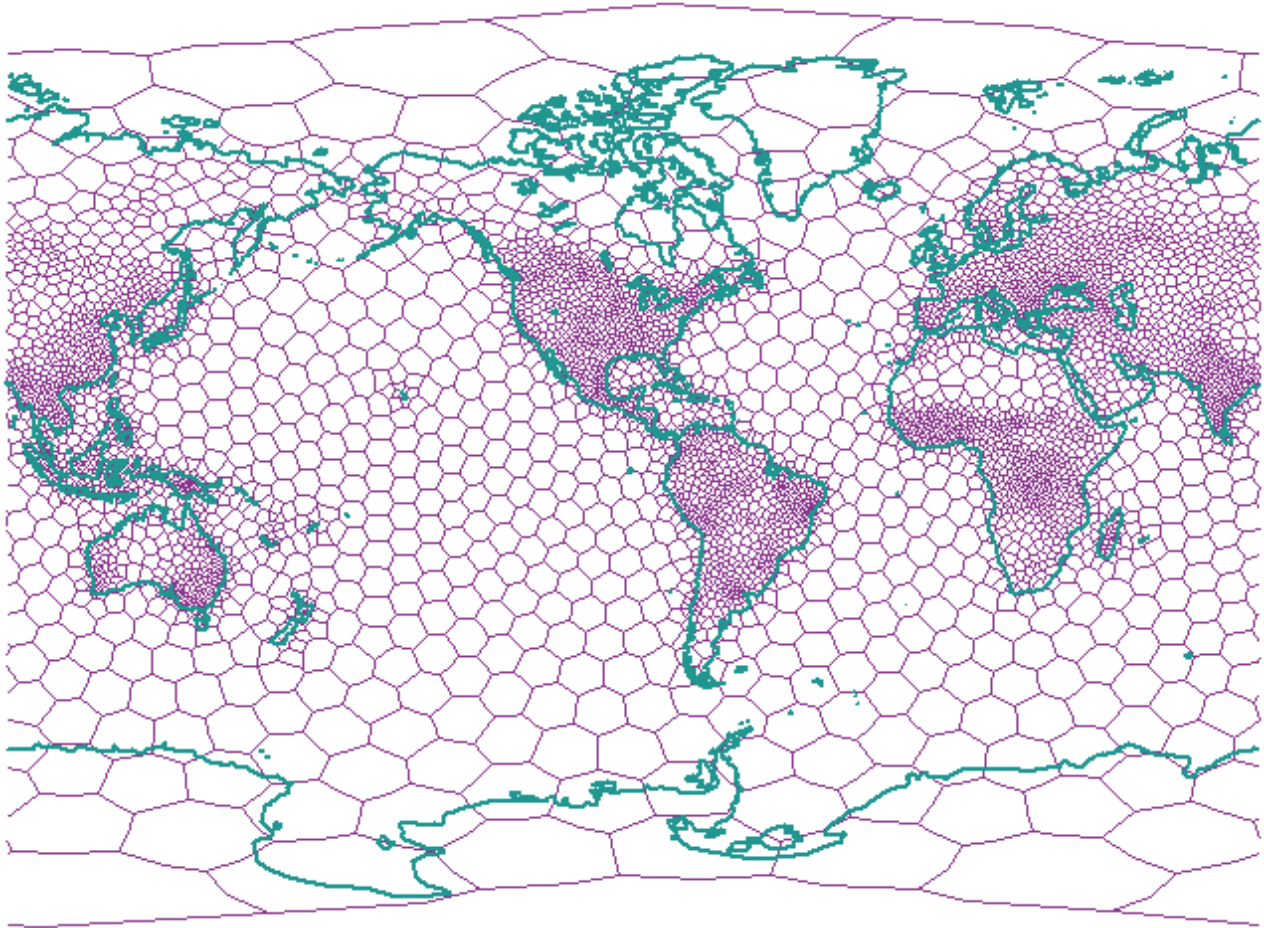


図 22. 世界全体 (の人口密度) を対象とするポロノイ・セル構造

米国 (ポロノイ ID: 2)

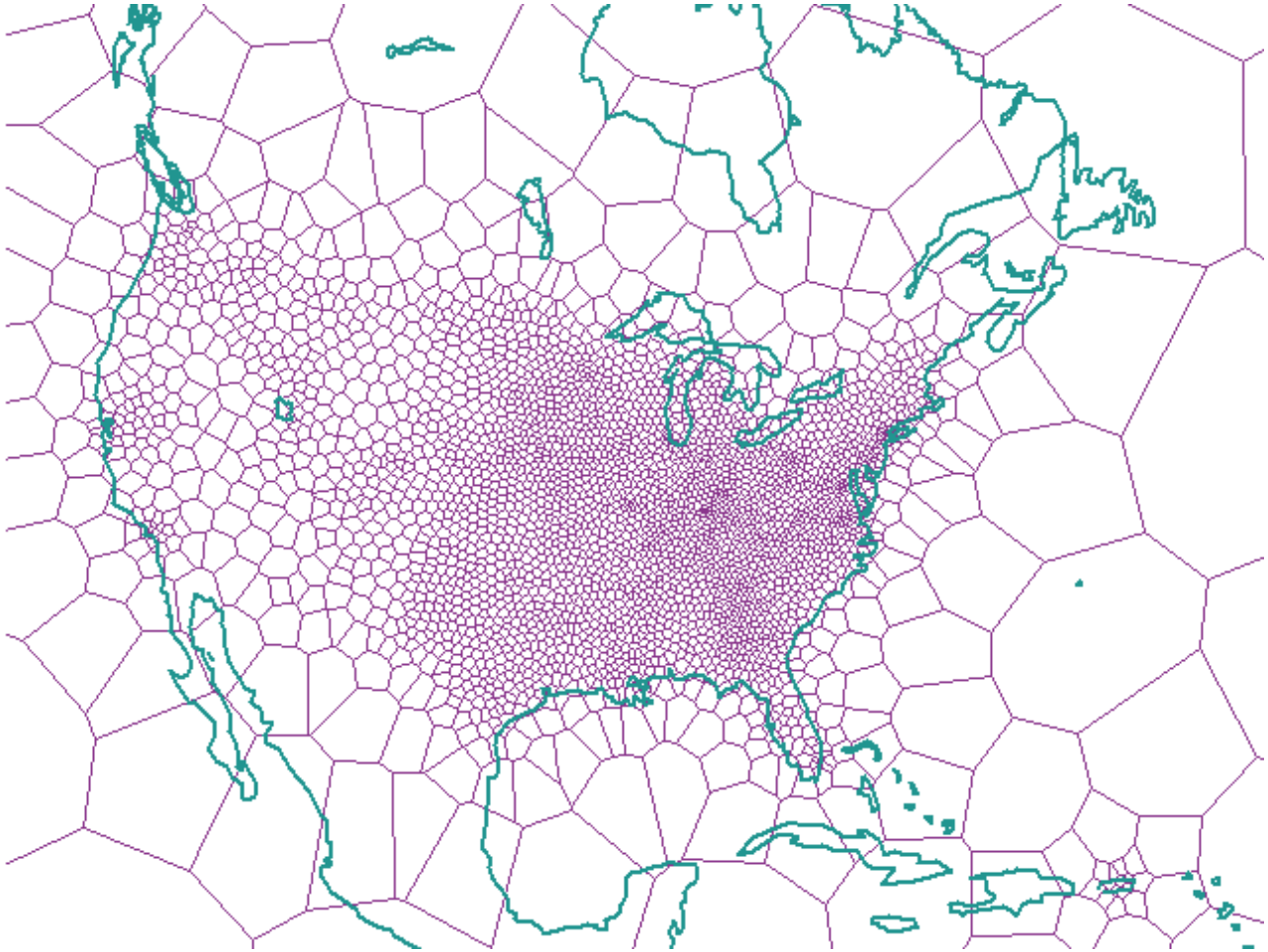


図 23. 米国を対象とするポロノイ・セル構造

カナダ (ボロノイ ID: 3)

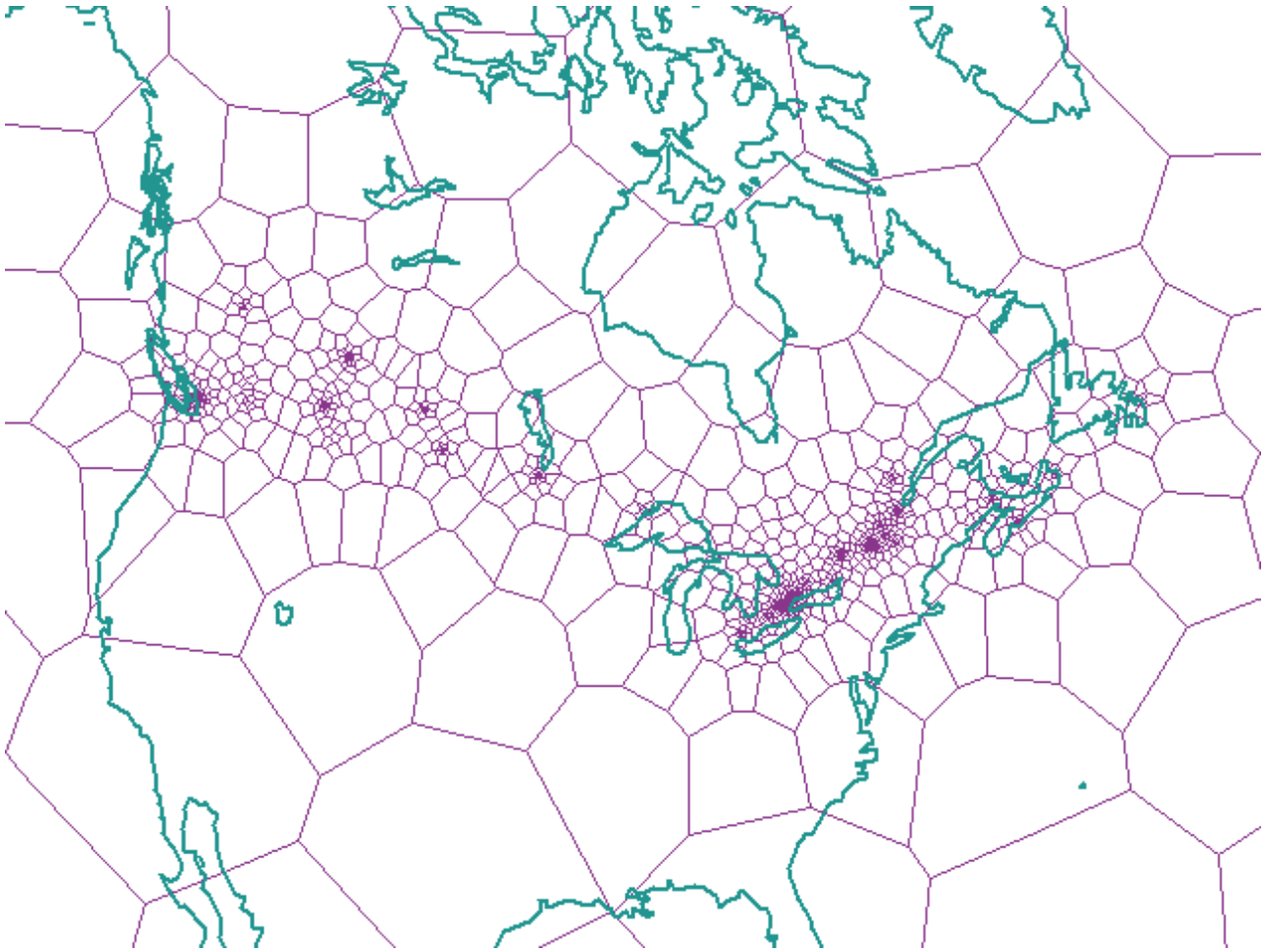


図 24. カナダを対象とするボロノイ・セル構造

インド (ポロノイ ID: 4)

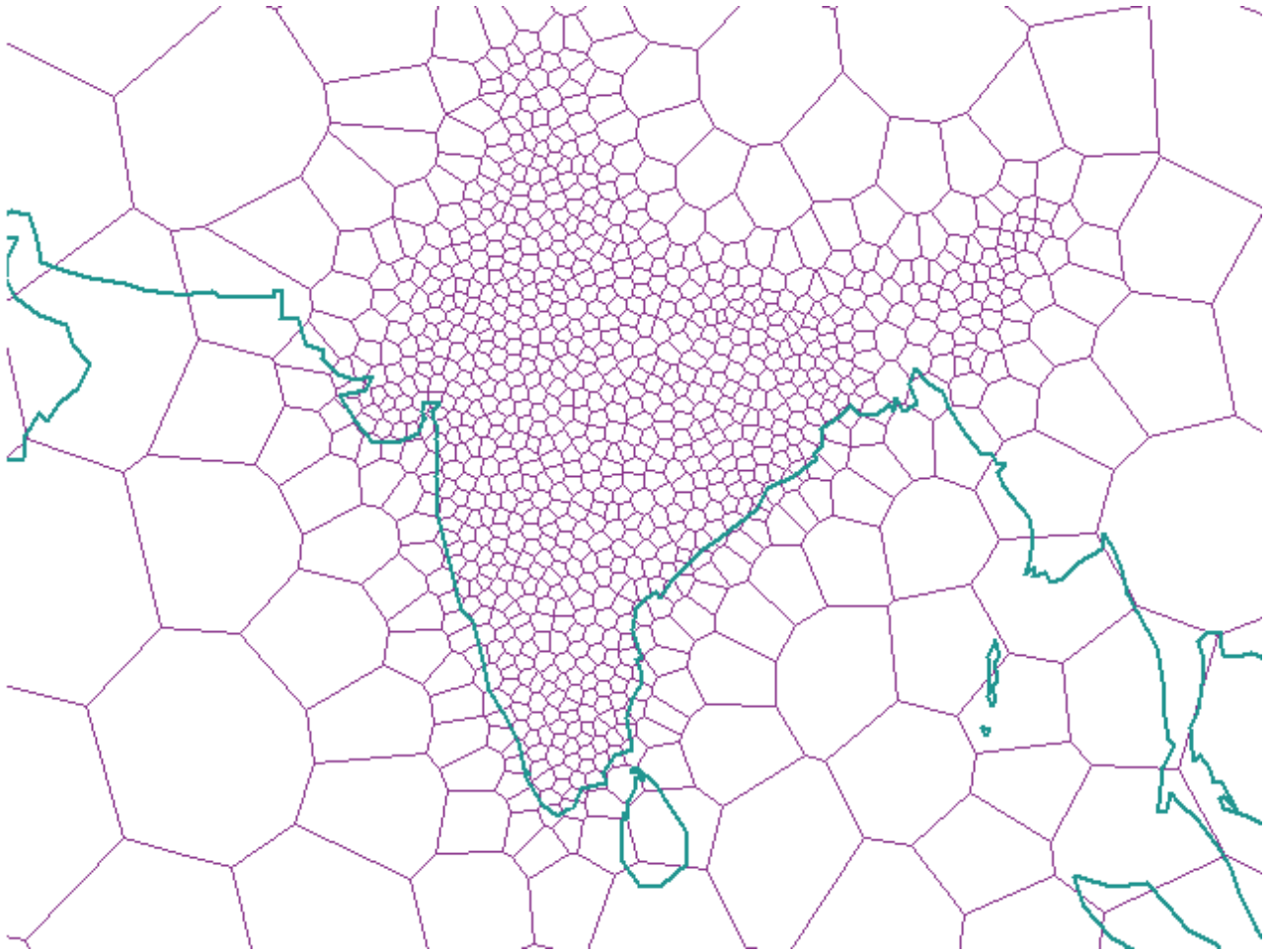


図 25. インドを対象とするポロノイ・セル構造

日本 (ポロノイ ID: 5)

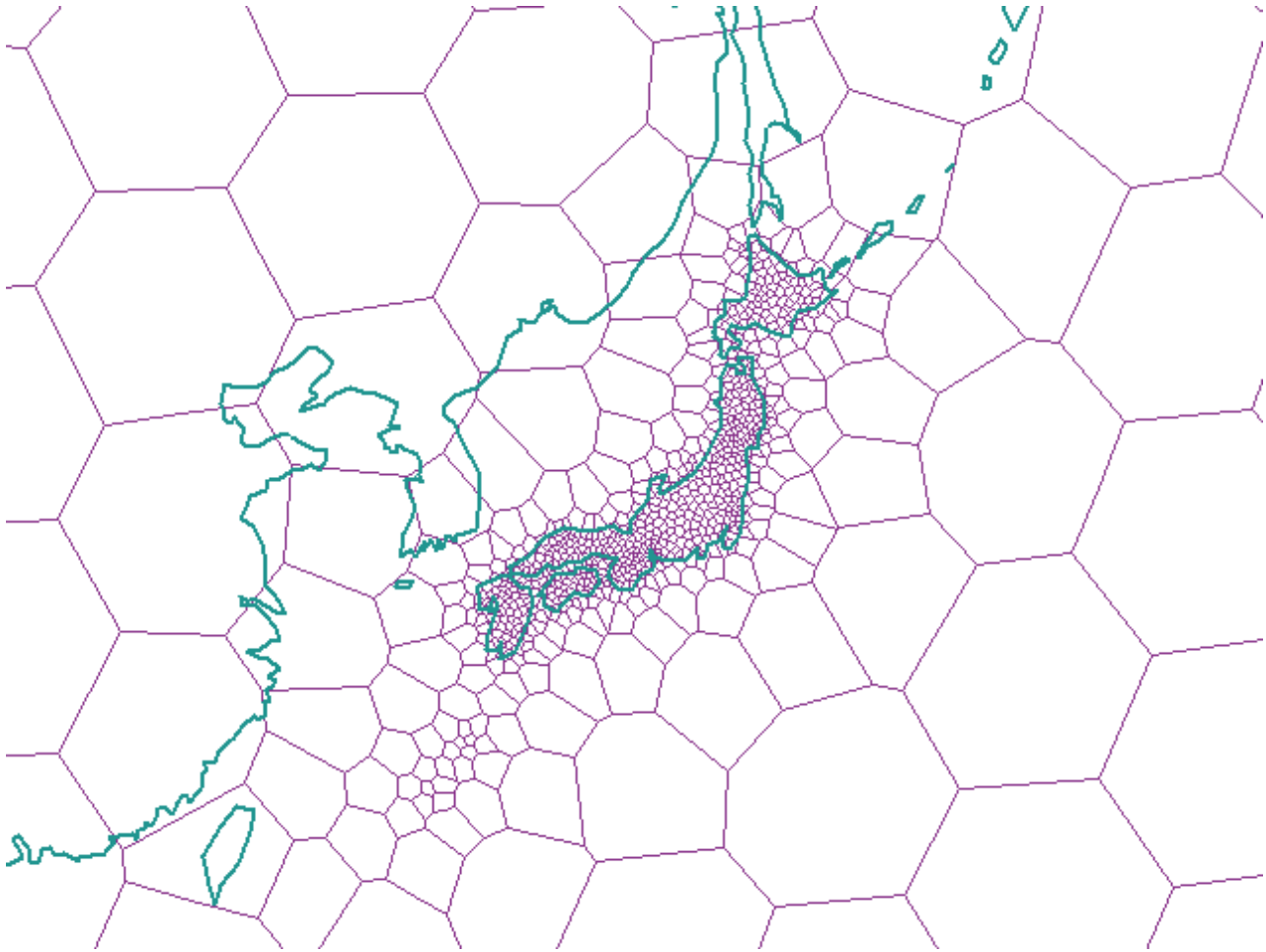


図 26. 日本を対象とするポロノイ・セル構造

アフリカ (ポロノイ ID: 6)

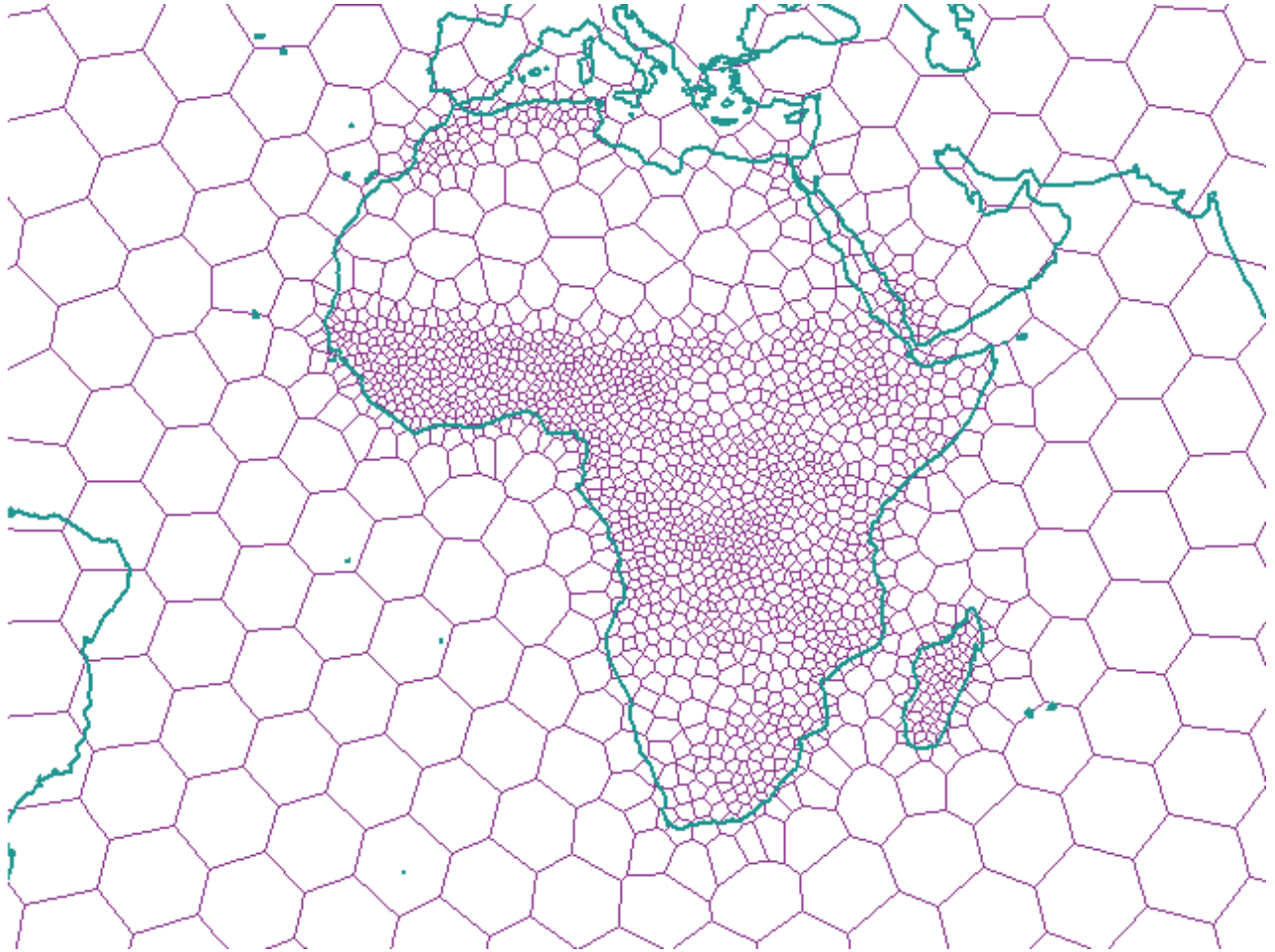


図 27. アフリカを対象とするポロノイ・セル構造

オーストラリア (ボロノイ ID: 7)

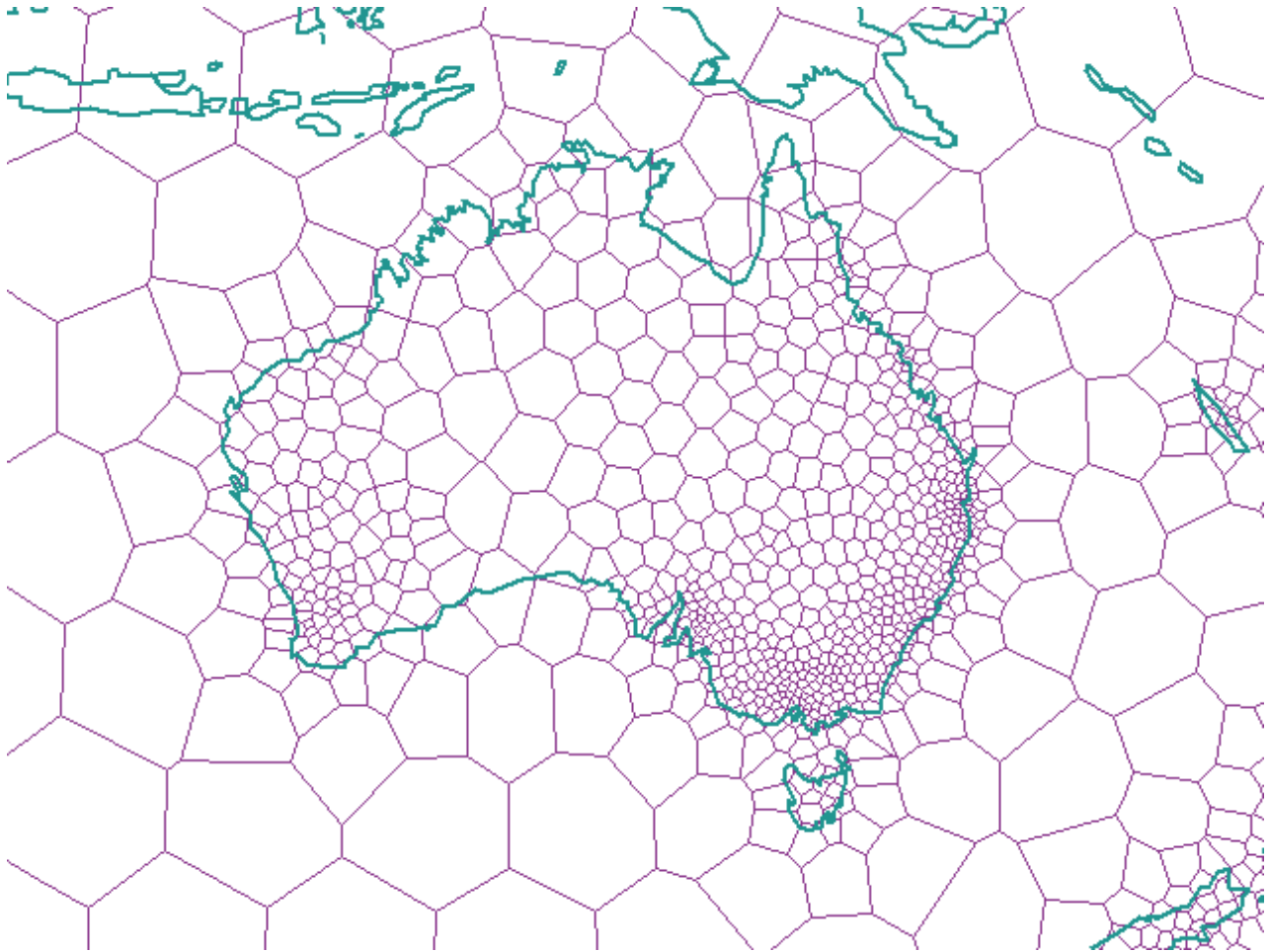


図 28. オーストラリアを対象とするボロノイ・セル構造

ヨーロッパ (ポロノイ ID: 8)

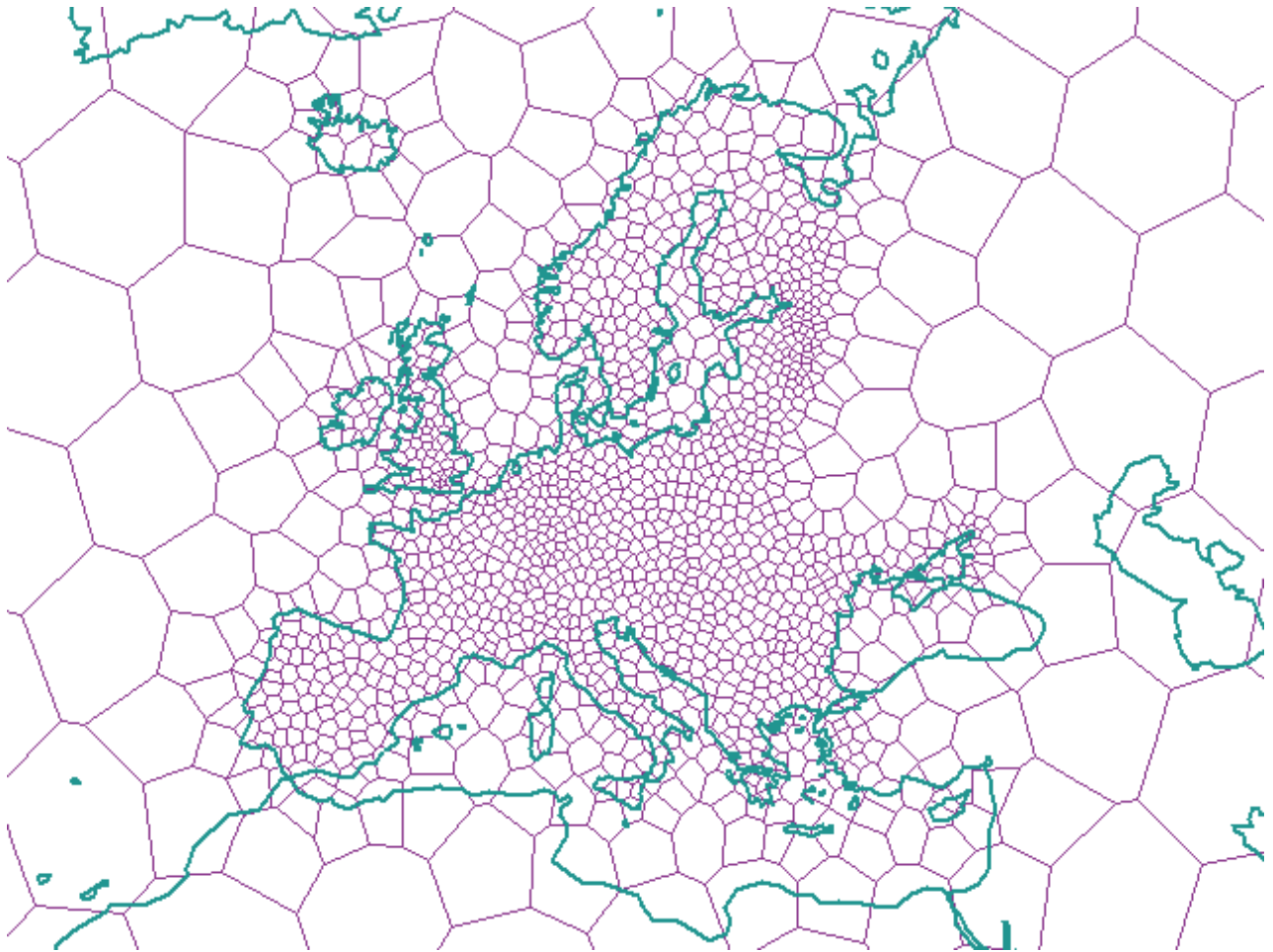


図 29. ヨーロッパを対象とするポロノイ・セル構造

北アメリカ (ポロノイ ID: 9)

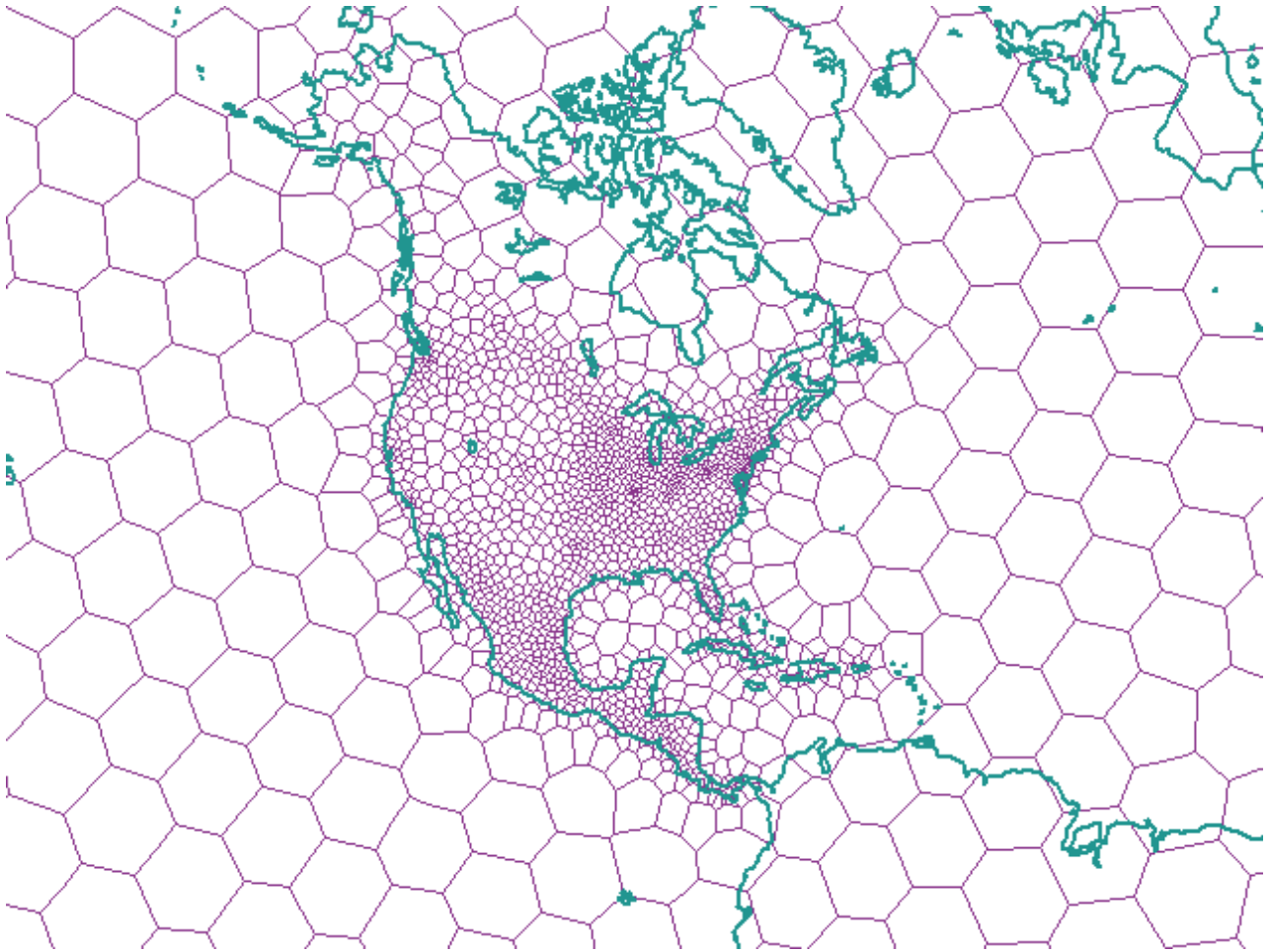


図 30. 北アメリカを対象とするポロノイ・セル構造

南アメリカ (ポロノイ ID: 10)



図 31. 南アメリカを対象とするポロノイ・セル構造

地中海 (ボロノイ ID: 11)

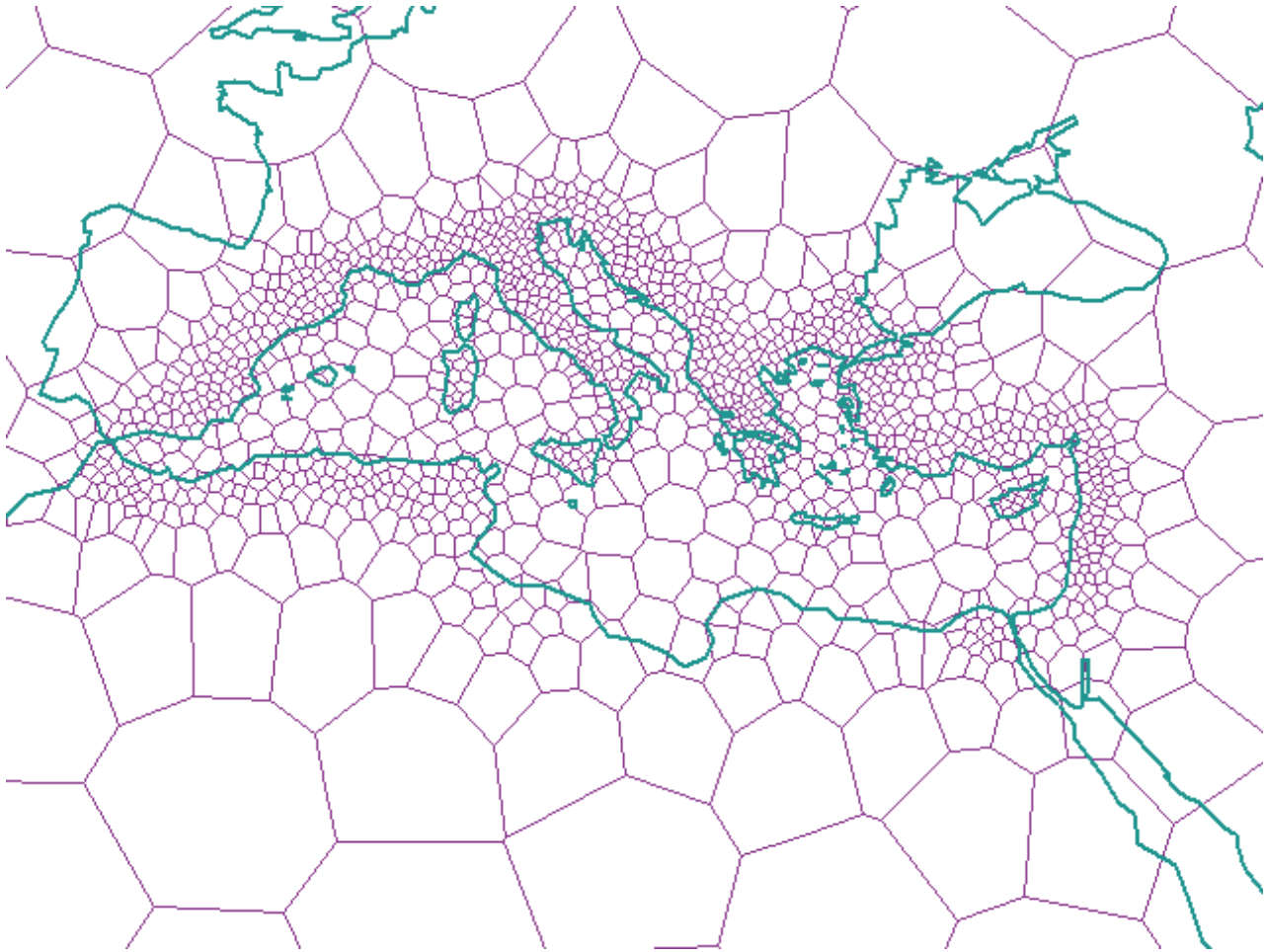


図 32. 地中海地域を対象とするボロノイ・セル構造

世界全体に均一にデータが分散、解像度は中程度 - dodeca04 (ポ
ロノイ ID: 12)



図 33. 世界全体を対象とするポロノイ・セル構造 (dodeca04)

世界の工業国を対象 - G7 諸国 (ボロノイ ID: 13)

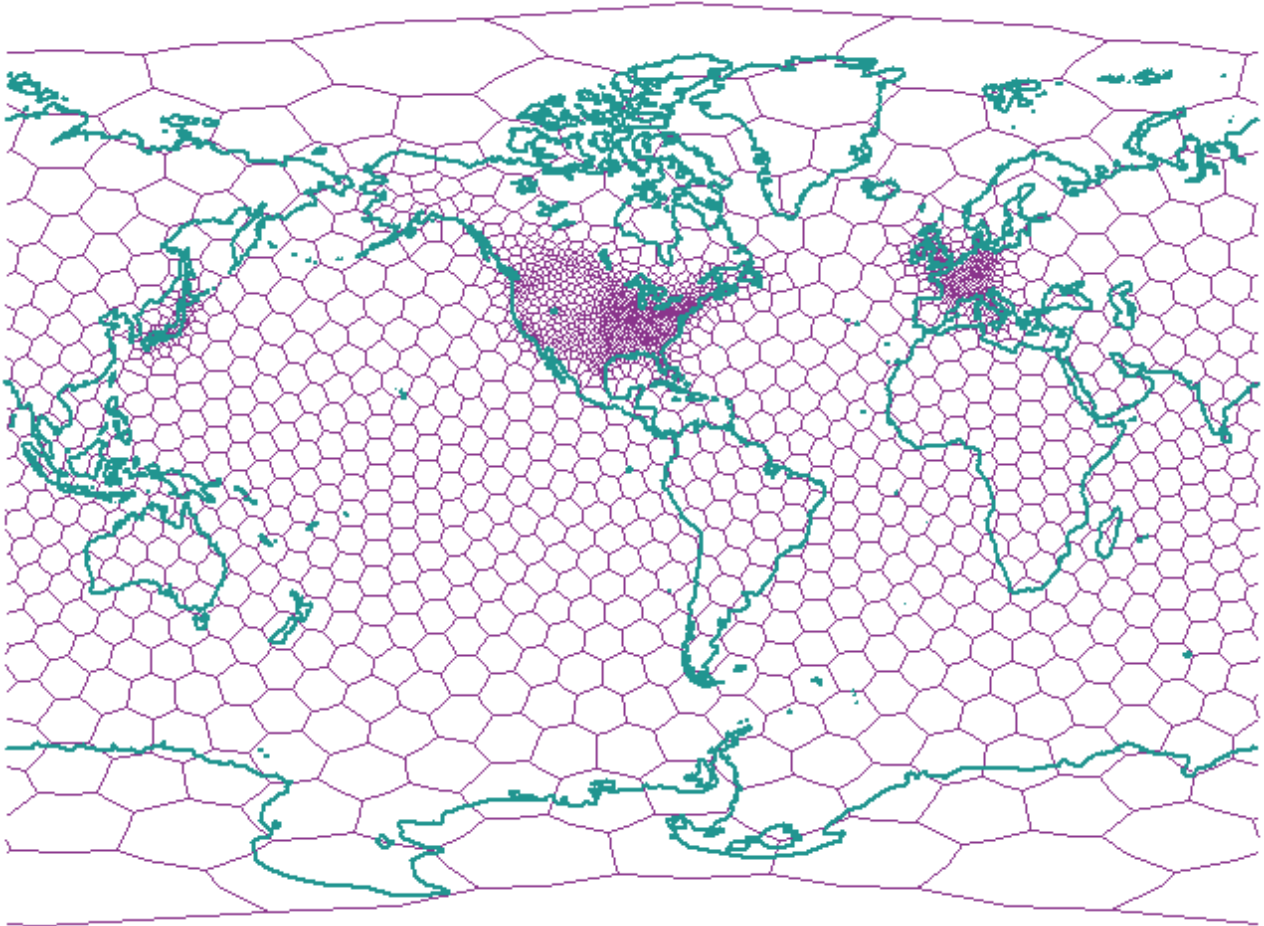


図 34. G7 諸国を対象とするボロノイ・セル構造

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)



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 Extender 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 Extender によってサポートされるストアード・プロシージャ、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 Extender は、データを測地ポロノイ・インデックス用にセル構造にまとめるために、最小外接長方形の代わりに、最小外接円 (MBC) を使用します。

測地図形の場合、MBC とは、図形を囲む円のことで、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。

ymin 外接円の *arc_radius*。

測地図形の場合、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、空間図形のものに似ていますが、関数の実行結果は、測地図形の場合、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 より大きくなる。たとえば、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 は -180、ST_MaxX は 180 になる。
- MBC が北極点を含む場合、ST_MaxY は 90 になる。
- MBC が南極点を含む場合、ST_MinY は -90 になる。

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 Extender では、中核となる技術が異なります。

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

- 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 Extender 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、極地や第 180 子午線などでゆがみが生じることのないようにします。

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、様々な状況での図形の扱い方も変わってきます。違いが生じるのは、たとえば、以下に示すようなことに関してです。

- 第 180 子午線をまたぐ直線セグメント (および測定距離)。
- 第 180 子午線をまたぐポリゴン。
- 第 180 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 Extender の特長としてまずあげられるのは、第 180 子午線をまたぐ図形や、極地に近い図形など、Spatial Extender の平面地球による表現では限界があった図形をうまく表現できるという点です。

第 180 子午線をまたぐ直線セグメント

211 ページの図 36 は、Spatial Extender と Geodetic Extender の、第 180 子午線をまたぐ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メジャーするために使用するものになっています。Geodetic Extender では、2 ポイント間の距離を、測地線、つまり楕円上の 2 ポイント間の最短経路に沿ってメジャーします (171 ページの『測地距離』を参照)。この場合の 2 点は、地球上のどこに配置してもかまいません。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 Extender は正しく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、第 -180 子午線は左端に、第 180 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用する必要があります。

- 直線セグメントを、第 180 子午線の東と西の 2 つの直線セグメントに分割する。
- 第 180 子午線が端にならないようデータの投影をし直す。

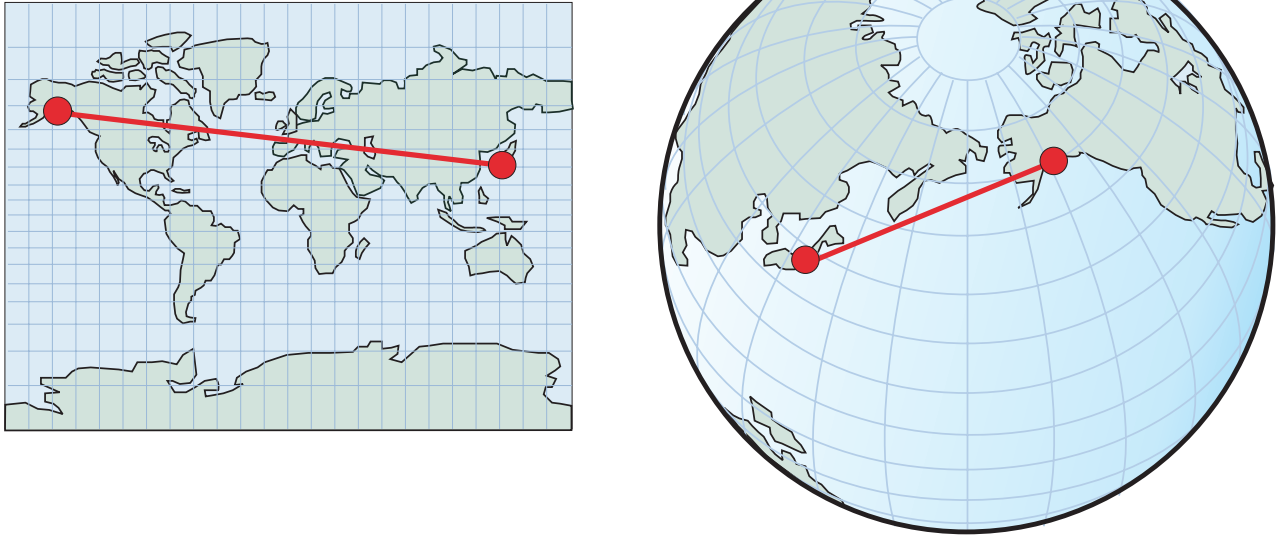


図 36. 第 180 子午線をまたぐ直線セグメント

第 180 子午線をまたぐポリゴン

第 180 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、以下のように、ポリゴンを第 180 子午線の東と西の部分に分割する必要があります。

```
MULTIPOLYGON(  
  ((-180 30, -165 30, -165 40, -180 30)),  
  ((180 30, 180 40, 165 40, 165 30, 180 30)))
```

212 ページの図 37 のように、球体地球表現 (Geodetic Extender) では、そのような分割は必要なく、1 つのポリゴンを手を加えずに使用できます。

```
POLYGON((-165 30, -165 40, 165 40, 165 30))
```

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

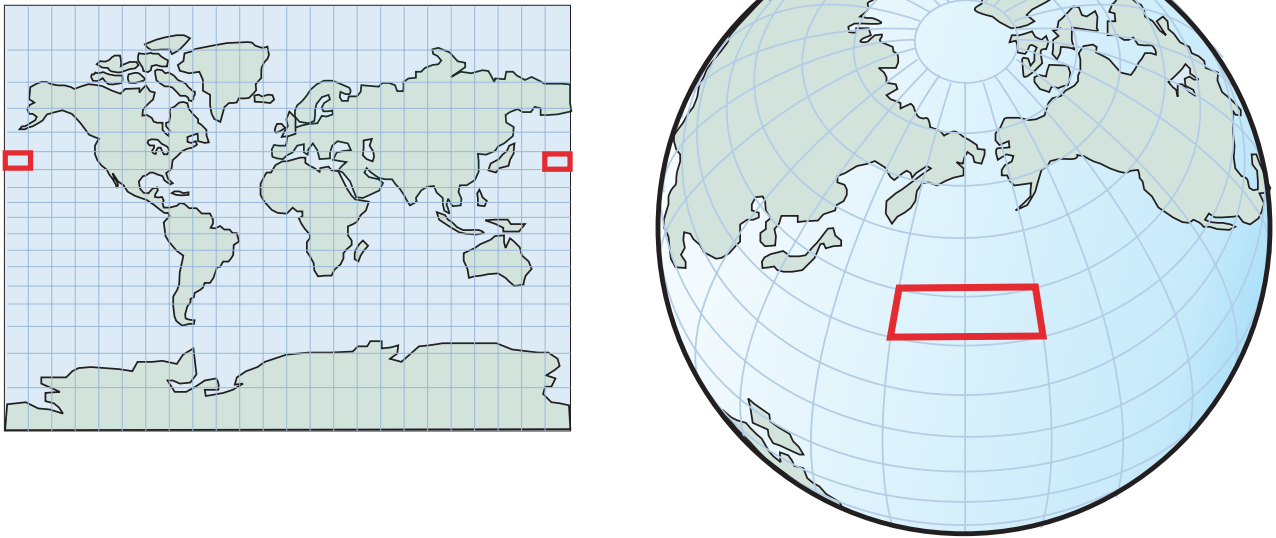


図 37. 第 180 子午線をまたぐポリゴン—ポリゴンの分割

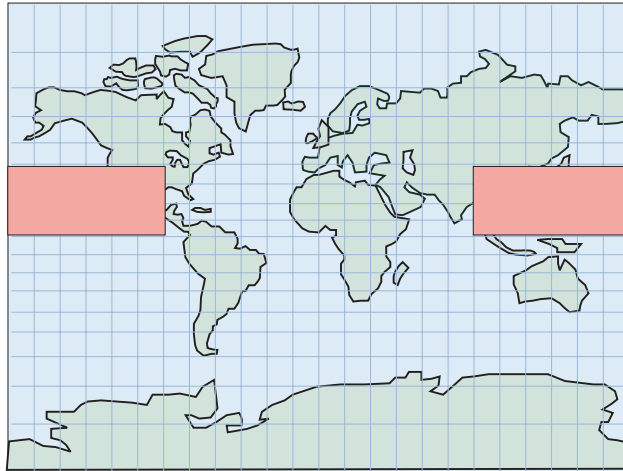
| Spatial Extender を使用しているにもかかわらずポリゴンを分割しなかった場合には、213 ページの図 38 に示すような別の領域を定義するため、ポリゴンの頂点の再配列が行われることになります。213 ページの図 38 の上側は、第 180 子午線をまたぐポリゴンの正しい配置を示しています。

| `POLYGON((90 0, -90 0, -90 40, 90 40, 90 0))`

| 213 ページの図 38 の下側は、頂点の再配列によって結果的に第 180 子午線をまたがず、第 0 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

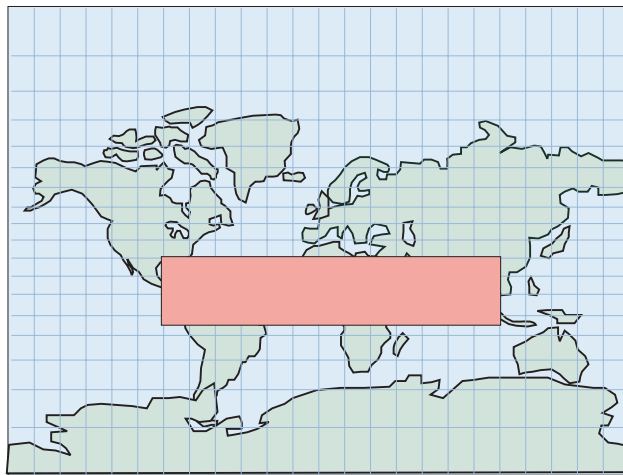


図 38. 第 180 子午線をまたぐポリゴン—頂点の再配列

再配列によってできるポリゴンが占める領域は、214 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ものになります。先述の
直線セグメントと同様、第 180 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。

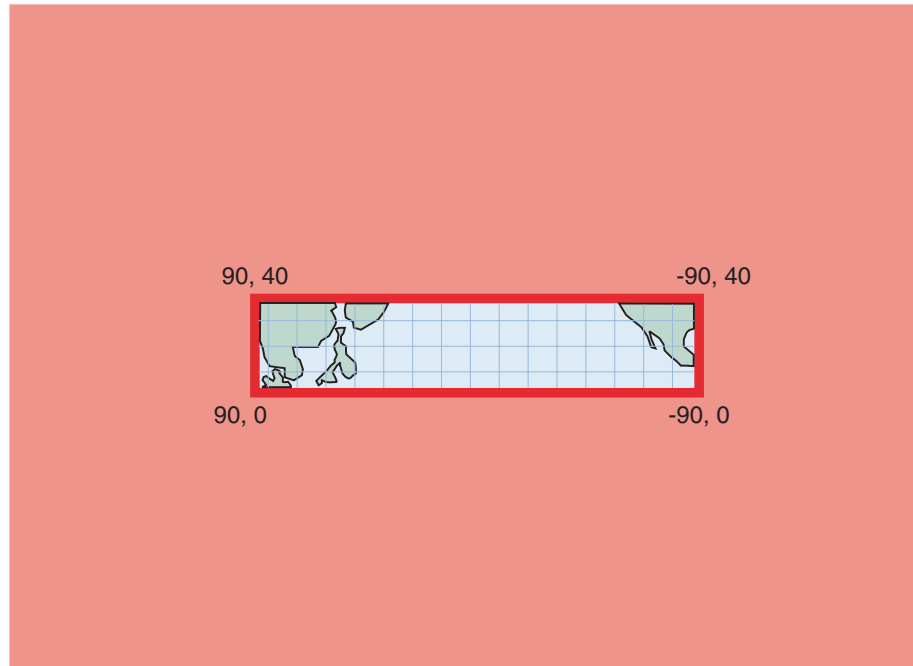


図39. 第 180 子午線をまたぐポリゴン—補完する領域

極点を囲むポリゴン

215 ページの図 40 は、Spatial Extender、Geodetic Extender で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。Spatial Extender の場合、平面地図の端の部分を抑うため、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。

```
POLYGON((-180 -90, 180 -90, 180 -60, -180 -60, -180 -90))
```

球体地球表現 (Geodetic Extender) では、南極を囲むポリゴンを、 -60° 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。

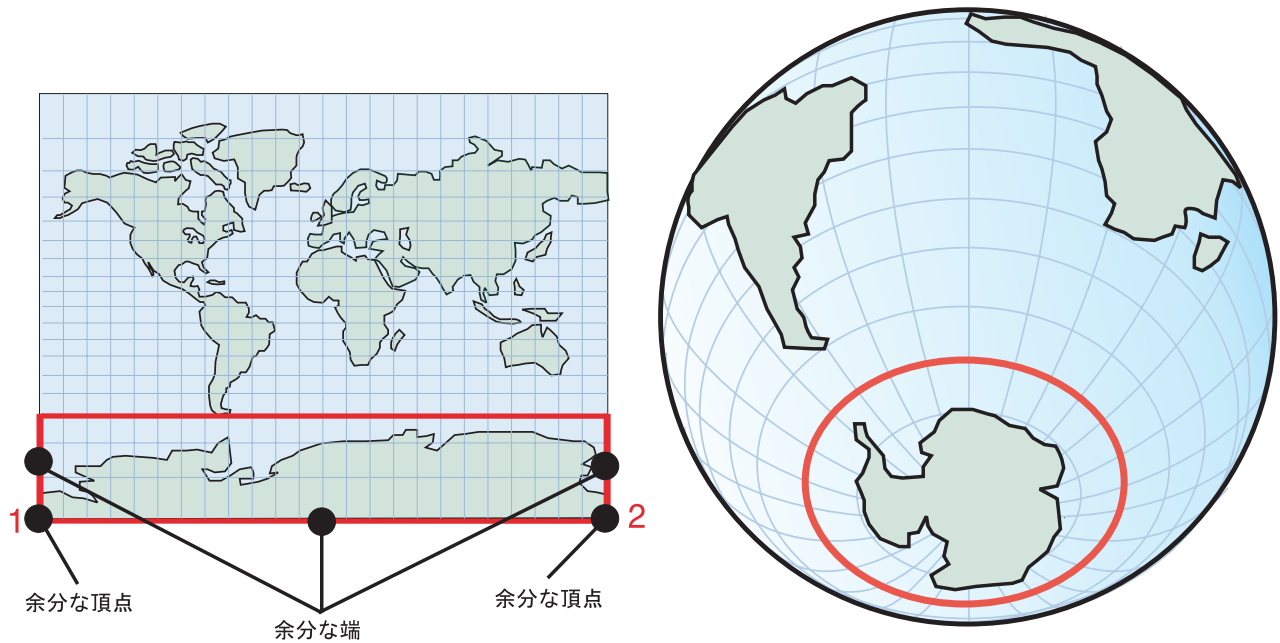


図 40. 極点を囲むポリゴン。

ここで取りあげた例の場合、適切な投影空間参照系を選択すれば、正しい結果が得られます。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。たとえば、第 180 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

半球、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利用する必要がある場合、Spatial Extender と Geodetic Extender では、対応のしかたが異なるという点に注意しなくてはなりません。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ても、正確な計算はできません。

たとえば、216 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 Extender) の両方で示したものです。

- 216 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第 180 子午線に沿ったリングを定義します。

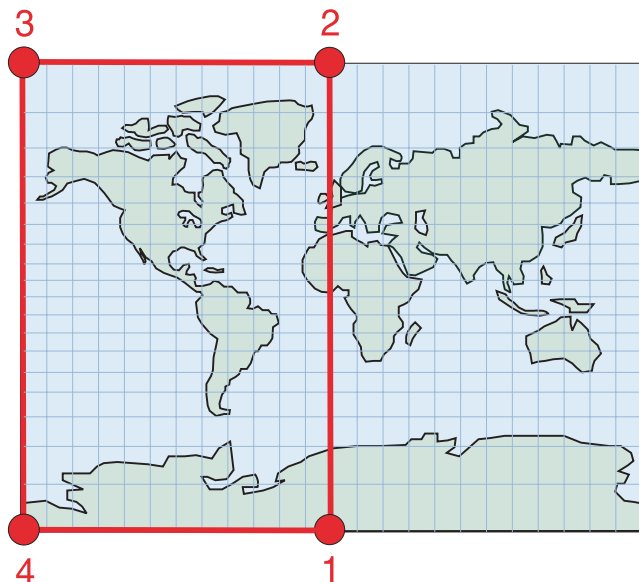
同じ 4 つのポイントを、反対の順序で指定すると、東半球が定義されます。

- 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。

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

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))

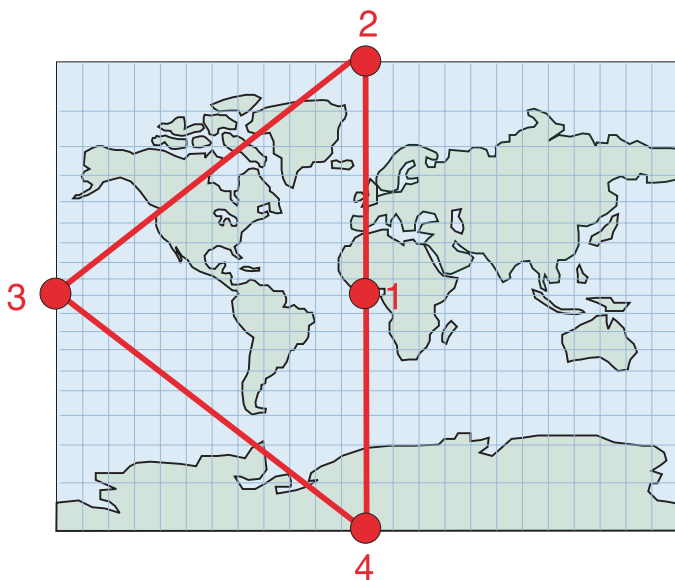


図 41. 半球を表すポリゴン。

217 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 Extender) で示したものです。

- 217 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 'POLYGON((180 -60, 180 60, -180 60, -180 -60, 180 -60))' で表したものです。
- 球体地球表現では、217 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。

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

```
'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。

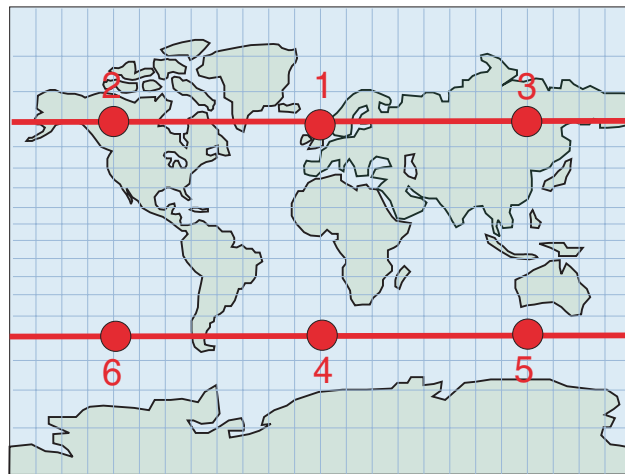
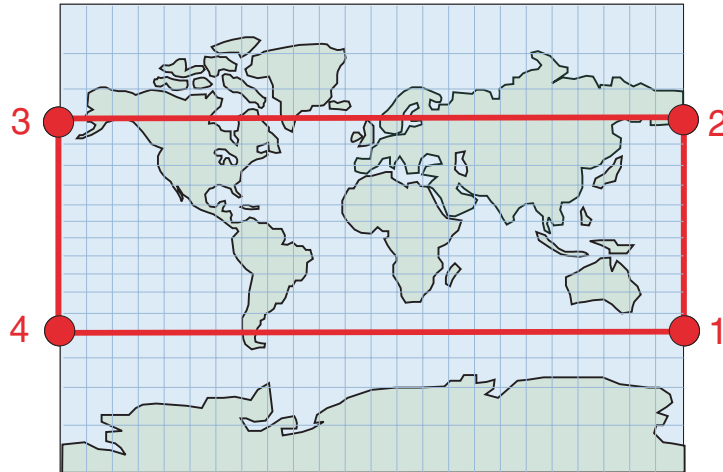


図 42. 赤道地帯を表すポリゴン

218 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 Extender) で示したものです。どちらの表現でも、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。

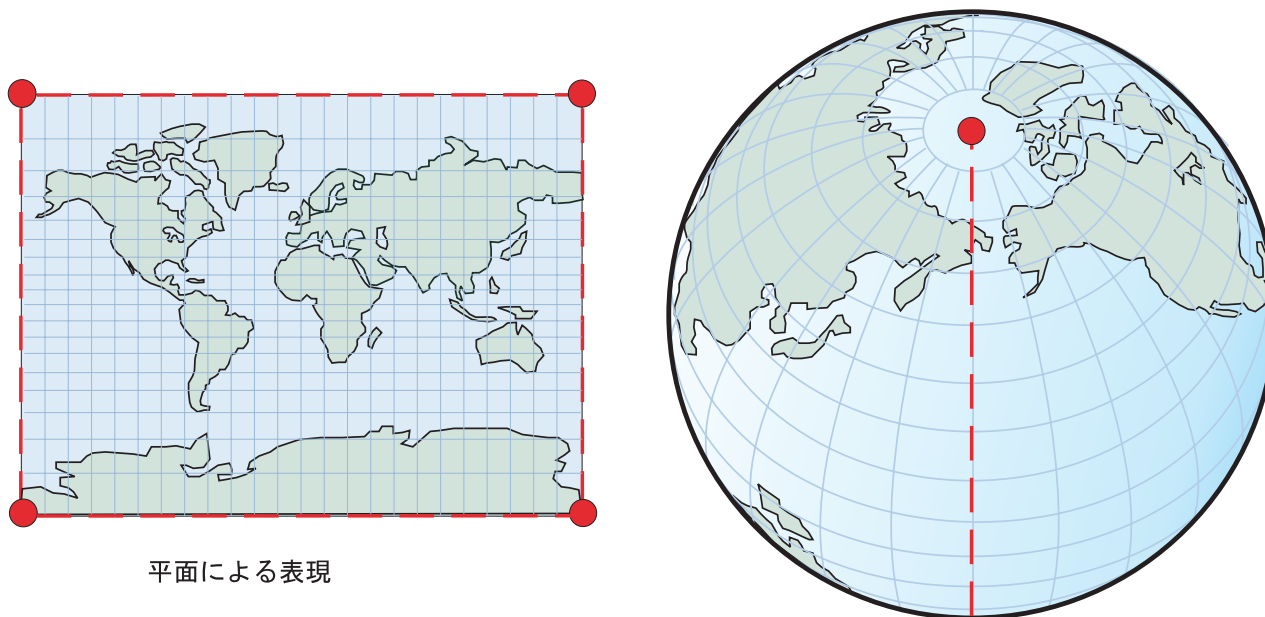


図 43. 地球全体を表すポリゴン

関連概念:

- 168 ページの『DB2 Geodetic Extender を使用すべき場合と DB2 Spatial Extender を使用すべき場合』
- 172 ページの『測地領域』
- 169 ページの『測地緯度と測地経度』
- 171 ページの『測地距離』
- 232 ページの『測地回転楕円体』

DB2 Geodetic Extender 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、DB2 Geodetic Extender は、Hipparchus 関数ライブラリーを基礎と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 Extender が Spatial Extender の関数うち、どれをサポートしているかをまとめたもので、Spatial Extender と Geodetic Extender で機能に違いがある場合は、それについても書いてあります。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 26. Geodetic Extender の関数サポート

関数	DB2 Geodetic Extender がサポートしているか?	DB2 Geodetic Extender での機能の変化
EnvelopesIntersect	サポートしている	なし
MBR 集約	サポートしていない	該当なし

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

表 26. *Geodetic Extender* の関数サポート (続き)

関数	DB2 Geodetic Extender がサポートしているか?	DB2 Geodetic Extender での機能の変化
ST_AppendPoint	サポートしていない	該当なし
ST_Area	サポートしている	デフォルトの測定単位がメートル。
ST_AsBinary	サポートしている	なし
ST_AsGML	サポートしている	なし
ST_AsShape	サポートしている	なし
ST_AsText	サポートしている	なし
ST_Boundary	サポートしていない	該当なし
ST_Buffer	サポートしている	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	サポートしていない	該当なし
ST_ChangePoint	サポートしていない	該当なし
ST_Contains	サポートしている	2 つの図形が同じ測地参照系 (SRS) で表現される必要がある。
ST_ConvexHull	サポートしていない	該当なし
ST_CoordDim	サポートしている	なし
ST_Crosses	サポートしていない	該当なし
ST_Difference	サポートしている	折れ線、複数折れ線に関してはサポートされていない。2 つの図形が同じ測地参照系 (SRS) で表現される必要がある。戻される図形のディメンションは、入力された図形と同じになる。
ST_Dimension	サポートしている	なし
ST_Disjoint	サポートしている	なし
ST_Distance	サポートしている	測地距離 を戻す。2 つの図形が同じ測地参照系 (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	サポートしている	なし
ST_Endpoint	サポートしている	なし
ST_Envelope	サポートしている	エンベロープは、図形の最小の外接円 (MBC) を囲むポリゴンになる。
ST_EnvIntersects	サポートしている	なし
ST_EqualCoordsys	サポートしている	なし
ST_Equals	サポートしていない	該当なし
ST_EqualSRS	サポートしている	なし
ST_ExteriorRing	サポートしている	なし
ST_FindMeasure または ST_LocateAlong	サポートしていない	該当なし
ST_Generalize	サポートしている	しきい値の単位がメートル。
ST_GeomCollection	サポートしていない	該当なし
ST_GeomCollFromTxt	サポートしていない	該当なし
ST_GeomCollFromWKB	サポートしていない	該当なし

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

表 26. Geodetic Extender の関数サポート (続き)

関数	DB2 Geodetic Extender がサポートしているか?	DB2 Geodetic Extender での機能の変化
ST_Geometry	サポートしている	なし
ST_GeometryN	サポートしている	なし
ST_GeometryType	サポートしている	なし
ST_GeomFromText	サポートしている	なし
ST_GeomFromWKB	サポートしている	なし
ST_GetIndexParams	サポートしていない	該当なし
ST_InteriorRingN	サポートしている	なし
ST_Intersection	サポートしている	戻される図形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、入力された図形のうちディメンションの低い方に合わされる。
ST_Intersects	サポートしている	2 つの図形が同じ測地参照系 (SRS) で表現される必要がある。
ST_Is3d	サポートしている	なし
ST_IsClosed	サポートしている	なし
ST_IsEmpty	サポートしている	なし
ST_IsMeasured	サポートしている	なし
ST_IsRing	サポートしていない	該当なし
ST_IsSimple	サポートしていない	該当なし
ST_IsValid	サポートしている	なし
ST_Length	サポートしている	デフォルトの測定単位がメートル。
ST_LineFromText	サポートしている	なし
ST_LineFromWKB	サポートしている	なし
ST_LineString	サポートしている	なし
ST_LineStringN	サポートしている	なし
ST_M	サポートしている	なし
ST_MaxM	サポートしている	なし
ST_MaxX	サポートしている	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	サポートしている	MBC の最大の Y 値を戻す。 注: MBC が北極点を含む場合、ST_MaxY は 90 になる。
ST_MaxZ	サポートしている	なし
ST_MBR	サポートしている	MBR は、図形の MBC を囲む図形。
ST_MBRIntersects	サポートしている	なし
ST_MeasureBetween または ST_LocateBetween	サポートしていない	該当なし
ST_MidPoint	サポートしている	なし

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

表 26. Geodetic Extender の関数サポート (続き)

関数	DB2 Geodetic Extender がサポートしているか?	DB2 Geodetic Extender での機能の変化
ST_MinM	サポートしている	なし
ST_MinX	サポートしている	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 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	サポートしている	MBC の最小の Y 値を戻す。 注: MBC が南極点を含む場合、ST_MinY は -90 になる。
ST_MinZ	サポートしている	なし
ST_MLineFromText	サポートしている	なし
ST_MLineFromWKB	サポートしている	なし
ST_MPointFromText	サポートしている	なし
ST_MPointFromWKB	サポートしている	なし
ST_MPolyFromText	サポートしている	なし
ST_MPolyFromWKB	サポートしている	なし
ST_MultiLineString	サポートしている	なし
ST_MultiPoint	サポートしている	なし
ST_MultiPolygon	サポートしている	なし
ST_NumGeometries	サポートしている	なし
ST_NumInteriorRing	サポートしている	なし
ST_NumLineStrings	サポートしている	なし
ST_NumPoints	サポートしている	なし
ST_NumPolygons	サポートしている	なし
ST_Overlaps	サポートしていない	該当なし
ST_Perimeter	サポートしている	デフォルトの測定単位がメートル。
ST_PerpPoints	サポートしていない	該当なし
ST_Point	サポートしている	なし
ST_PointFromText	サポートしている	なし
ST_PointFromWKB	サポートしている	なし
ST_PointN	サポートしている	なし
ST_PolyFromText	サポートしている	なし
ST_PolyFromWKB	サポートしている	なし
ST_PointOnSurface	サポートしている	なし
ST_Polygon	サポートしている	なし
ST_PolygonN	サポートしている	なし
ST_Relate	サポートしていない	該当なし
ST_RemovePoint	サポートしていない	該当なし
ST_SrsId or ST_SRID	サポートしている	なし

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

表 26. Geodetic Extender の関数サポート (続き)

関数	DB2 Geodetic Extender がサポートしているか?	DB2 Geodetic Extender での機能の変化
ST_SrsName	サポートしている	なし
ST_StartPoint	サポートしている	なし
ST_SymDifference	サポートしている	折れ線、複数折れ線に関してはサポートされていない。戻される図形のディメンションは、入力された図形と同じになる。2 つの図形が同じ測地参照系 (SRS) で表現される必要がある。
ST_ToGeomColl	サポートしていない	該当なし
ST_ToLineString	サポートしている	なし
ST_ToMultiLine	サポートしている	なし
ST_ToMultiPoint	サポートしている	なし
ST_ToPoint	サポートしている	なし
ST_ToPolygon	サポートしている	なし
ST_Touches	サポートしていない	該当なし
ST_Transform	サポートしている	なし。注：座標変換はポイントごとに行われる。変換を測地座標系と非投影平面座標系の間で行う場合には、ポリゴンや折れ線の中に、第 180 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 Extender では、そのようなケースへの対処が異なるので、平面地球座標系では妥当であった図形が、球体地球では妥当でなくなる (あるいはその逆) 恐れもある。詳細については、209 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	サポートしている	2 つの図形が同じ測地参照系 (SRS) で表現される必要がある。
ST_Within	サポートしている	2 つの図形が同じ測地参照系 (SRS) で表現される必要がある。
ST_WKBToSQL	サポートしている	なし
ST_WKTToSQL	サポートしている	なし
ST_X	サポートしている	なし
ST_Y	サポートしている	なし
ST_Z	サポートしている	なし
和集約	サポートしていない	該当なし

関連概念:

- 168 ページの『DB2 Geodetic Extender を使用すべき場合と DB2 Spatial Extender を使用すべき場合』
- 172 ページの『測地領域』
- 169 ページの『測地緯度と測地経度』

- 171 ページの『測地距離』

関連タスク:

- 189 ページの『測地ポロノイ・インデックスの作成』

関連資料:

- 209 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』

DB2 Geodetic Extender のストアード・プロシージャとカタログ・ビュー

DB2 Geodetic Extender 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・プロシージャの一部をサポートしています。

ただし、Geodetic Extender は以下のストアード・プロシージャ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 Extender は、DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カatalog・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。提供されている測地参照系の完全なリストは、『DB2 Geodetic Extender によってサポートされる測地系』に記載されています。

DB2 Geodetic Extender によってサポートされる測地系

62 ページの『測地座標系』で触れたとおり、*測地* 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。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、個々に固有の空間参照系 ID (SRID) をもっています。表 28 は、DB2 Geodetic Extender 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因数は、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 27. 定義済み測地 SRS のオフセット値、あるいはスケール因数の値

SRS パラメーター	値
<i>xOffset</i>	-180
<i>yOffset</i>	-90
<i>zOffset</i>	-50000
<i>mOffset</i>	-1000
<i>xScale</i>	5965232
<i>yScale</i>	5965232

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

表 27. 定義済み測地 SRS のオフセット値、あるいはスケール因数の値 (続き)

SRS パラメーター	値
<i>zScale</i>	1000
<i>mScale</i>	1000

yScale は常に *xScale* と同じです。

ユーザーは、自らの空間参照系に対応するものとして、表 28 に示したどの測地系でも選択できます。ただ、できればデータに最も適合したものを選択するにすべきです。たとえば、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、対応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 28. 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

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

表 28. 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
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
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

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

表 28. 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
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
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)

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

表 28. 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
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

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

表 28. 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
2000000136	Hungarian Datum 1972	GRS 1967
2000000137	Hu Tzu Shan	International 1924
2000000138	Indian 1954	Everest Adjustment 1937
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

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

表 28. 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
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
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

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

表 28. 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
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	Observatorio	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)
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

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

表 28. 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
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	球面
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

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

表 28. 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
2000000288	S-JTSK (Ferro)	Bessel 1841
2000000289	Tananarive 1925	International 1924
2000000290	Tananarive 1925 (Paris)	International 1924
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測地座標系の構成要素です。

DATUM で座標系を定義する際には、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]]
```

Spatial Extender と Geodetic Extender の提供する回転楕円体のリストは、535 ページの『サポートされる座標系』に記載されています。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、Supported spheroids 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。

関連概念:

- 62 ページの『測地座標系』

関連資料:

- 297 ページの『DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビュー』
- 248 ページの『ST_create_coordsys』

第 5 部 参照資料

第 20 章 ストアード・プロシージャ

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する、ストアード・プロシージャの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・プロシージャが暗黙的に呼び出されます。たとえば、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するストアード・プロシージャを呼び出します。

あるいは、アプリケーション・プログラム内で明示的にストアード・プロシージャを呼び出すことができます。

データベースに DB2 Spatial Extender ストアード・プロシージャを呼び出す場合、ほとんどは、事前に DB2 コントロール・センターを使用するかまたは、ST_enable_db ストアード・プロシージャを直接呼び出して、そのデータベースに地理情報操作を行えるようにしておく必要があります。(このストアード・プロシージャを呼び出す方法については、このセクションの後にある ST_enable_db のトピックに記述されています。)

データベースに地理情報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・プロシージャを説明しています。

- GSE_export_sde
- GSE_import_sde
- ST_alter_coordsys
- ST_alter_srs
- ST_create_coordsys
- ST_create_srs
- ST_disable_autogeocoding
- ST_disable_db
- ST_drop_coordsys
- ST_drop_srs
- ST_enable_autogeocoding
- ST_enable_db
- ST_export_shape
- ST_import_shape
- ST_register_geocoder
- ST_register_spatial_column

ストアド・プロシージャ

- ST_remove_geocoding_setup
- ST_run_geocoding
- ST_setup_geocoding
- ST_unregister_geocoder
- ST_unregister_spatial_column

ストアド・プロシージャ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。

GSE_export_sde

このストアド・プロシージャは、地理情報列とこの列に関連する表を SDE 転送ファイルにエクスポートするために使用します。

制約事項:

- 表またはビューに存在しなければならない地理情報列は 1 つだけです。
- 地理情報列は登録されている必要があります。
- 既存の SDE ファイルに付加することはできません。

許可:

このストアド・プロシージャを呼び出すユーザー ID は、SYSADM または DBADM 権限のいずれかを持つ必要があります。さらにこのユーザー ID は、エクスポートされる表の SELECT 特権を持つ必要があります。

構文:

```
▶▶ db2gse.GSE_export_sde ( ( table_schema , table_name , column_name )  
                        | NULL |  
▶▶ , file_name , ( where_clause )  
                        | NULL |
```

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

エクスポートする表の、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。

column_name

エクスポートする、登録された地理情報列の名前。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。

file_name

指定した地理情報列およびこれに関連する表をエクスポートする、SDE 転送ファイルの名前。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。

このパラメーターのデータ・タイプは VARCHAR(256) です。

where_clause

SQL WHERE 文節の本体を指定し、これにより、エクスポートするレコードのセットに対する制限を定義します。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 文節に制限は定義されません。

このパラメーターを指定する場合は、エクスポートする表内の任意の属性列を参照することができます。

このパラメーターのデータ・タイプは VARCHAR(1024) です。

出力パラメーター:*msg_code*

ストアード・プロシージャ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

プロシージャから戻される、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を使用して、GSE_export_sde ストアード・プロシージャ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 という名前の表から SDE ファイルにデータをエクスポートします。

```
call db2gse.GSE_export_sde(NULL,'CUSTOMERS','LOCATION','/tmp/export_sde_file',
    NULL,?,?)
```

GSE_export_sde

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・プロシージャの実行後に表示されます。

関連資料:

- 240 ページの『GSE_import_sde』

GSE_import_sde

このストアード・プロシージャは、地理情報操作が使用可能になっているデータベースに、SDE 転送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・プロシージャは、次の 2 つの方法のいずれかで操作することができます。

- SDE 転送ファイルの宛先が、登録された地理情報列を持つ既存の表の場合、DB2 Spatial Extender はファイルのデータを表にロードします。
- 上記以外の場合、DB2 Spatial Extender は、地理情報列を持つ表を作成し、この列を登録し、地理情報列および表のその他の列にファイルのデータをロードします。

SDE 転送ファイルに指定されている空間参照系は、DB2 Spatial Extender に登録されている空間参照系と比較されます。指定されたシステムが登録されたシステムと一致した場合、転送データ内のすべてのデータ値は、ロード時に、登録されたシステムが指定する方法で変更されます。指定されたシステムが、登録されたシステムのどれとも一致しない場合、DB2 Spatial Extender は、変更を指定するために新しい空間参照系を作成します。

許可:

既存の表にデータをインポートする場合、このストアード・プロシージャ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。

- データがインポートされる表を含むデータベースに対する SYSADM 権限または DBADM 権限
- この表に対する CONTROL 特権

データをインポートする表を作成する必要がある場合、このストアード・プロシージャを呼び出すユーザー ID は、作成される表を含むデータベースに対して、SYSADM または DBADM 権限のいずれかを持つ必要があります。

構文:

```
▶▶ db2gse.GSE_import_sde ( ( table_schema , table_name , column_name )
                             └── NULL ─── )
▶▶ , file_name , ( commit_scope )
                             └── NULL ─── ) ▶▶
```

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

SDE 転送データをロードする表の、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。

column_name

SDE 転送ファイルの空間データのロード先である、登録された列の名前。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。

file_name

インポートされる SDE 転送ファイルの名前。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。

このパラメーターのデータ・タイプは VARCHAR(256) です。

commit_scope

レコードをいくつインポートしたら COMMIT を発行するかを示すレコード数を指定する。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。

出力パラメーター:

msg_code

ストアード・プロシージャ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

プロシージャから戻される、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を使用して、GSE_import_sde ストアド・プロシージャ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/customerSDE という名前の SDE ファイルを CUSTOMERS という名前の表にインポートします。この CALL コマンドは、5 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ています。

```
call db2gse.GSE_import_sde(NULL,'CUSTOMERS','LOCATION',
    '/tmp/customerSde', 5, ?,?)
```

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、ストアド・プロシージャの実行後に表示されます。

関連資料:

- 238 ページの『GSE_export_sde』

ST_alter_coordsys

このストアド・プロシージャは、データベース内の座標系定義を更新するために使用します。このストアド・プロシージャが処理されると、座標系についての情報が DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューで更新されます。

注意: このストアド・プロシージャの使用には注意が必要です。このストアド・プロシージャを使用して座標系の定義を変更した場合、この座標系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:

このストアド・プロシージャを呼び出すユーザー ID は、SYSADM または DBADM 権限のいずれかを持つ必要があります。

構文:

```
▶ db2gse.ST_alter_coordsys(—coordsys_name—, —definition—, —
    NULL—
    —organization—, —organization_coordsys_id—, —description—)▶
    NULL NULL NULL
```

パラメーターの説明:*coordsys_name*

座標系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

座標系を定義します。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、座標系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、座標系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ると座標系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団体により、この座標系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、その団体の座標系 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ると、座標系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、座標系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

プロシージャから戻される、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。

ST_alter_coordsys

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を使用して、ST_alter_coordsys ストアド・プロシージャ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 という名前の座標系を変更します。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、ストアド・プロシージャの実行後に表示されます。

関連資料:

- 248 ページの『ST_create_coordsys』
- 260 ページの『ST_drop_coordsys』

ST_alter_srs

このストアド・プロシージャは、データベース内の空間参照系を更新するために使用します。このストアド・プロシージャが処理されると、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、地理情報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地理情報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。

注意: このストアド・プロシージャの使用には注意が必要です。このストアド・プロシージャを使用して、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:

このストアド・プロシージャを呼び出すユーザー ID は、SYSADM または DBADM 権限のいずれかを持つ必要があります。

構文:

```
►►—db2gse.ST_alter_srs—(—srs_name—, —srs_id—, —x_offset—, —————→  
                          NULL                          NULL
```

```

└─ x_scale ─┘, └─ y_offset ─┘, └─ y_scale ─┘, └─ z_offset ─┘,
  NULL      NULL      NULL      NULL
└─ z_scale ─┘, └─ m_offset ─┘, └─ m_scale ─┘, └─ coordsys_name ─┘,
  NULL      NULL      NULL      NULL
└─ description ─┘)
  NULL

```

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図形の、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。

図形が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール因数 *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図形の、すべての X 座標のスケール因数を指定します。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。

図形が外部表記 (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、スケール因数が適用されます (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図形の、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。

図形が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール因数 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図形の、すべての Y 座標のスケール因数を指定します。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。

図形が外部表記 (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、スケール因数が適用されます (乗算)。このスケール因数は 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図形の、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。

図形が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール因数 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図形の、すべての Z 座標のスケール因数を指定します。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。

図形が外部表記 (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、スケール因数が適用されます (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図形の、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。

図形が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール因数 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図形の、すべての M 座標のスケール因数を指定します。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。

図形が外部表記 (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、スケール因数が適用されます (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系を一意的に識別します。座標系は、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系は変更されません。

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

プロシージャから戻される、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を使用して、ST_alter_srs ストアード・プロシージャ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,'SRS for GSE Demo Program: offices table',?,?)
```

ST_alter_srs

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、ストアド・プロシージャの実行後に表示されます。

関連資料:

- 261 ページの『ST_drop_srs』
- 250 ページの『ST_create_srs』

ST_create_coordsys

このストアド・プロシージャは、新しい座標系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。このストアド・プロシージャが処理されると、座標系についての情報が DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューに追加されます。

許可:

このストアド・プロシージャを呼び出すユーザー ID は、SYSADM または DBADM 権限のいずれかを持つ必要があります。

構文:

```
db2gse.ST_create_coordsys(—coordsys_name—, —definition—, —————→  
organization, organization_coordsys_id, description)————→  
NULL NULL NULL
```

パラメーターの説明:

coordsys_name

座標系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

座標系を定義します。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、座標系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系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、すべての座標系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、座標系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、座標系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

プロシージャから戻される、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を使用して、ST_create_coordsys ストアード・プロシージャ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系を作成します。

- *coordsys_name* パラメーター: NORTH_AMERICAN_TEST
- *definition* パラメーター:

ST_create_coordsys

```
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、ストアド・プロシージャの実行後に表示されます。

関連資料:

- 261 ページの『ST_drop_srs』
- 244 ページの『ST_alter_srs』

ST_create_srs

このストアド・プロシージャは、空間参照系を作成するために使用します。空間参照系は、座標系、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、地理情報操作のパフォーマンスも大幅に改善できます。

このストアド・プロシージャには 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因数）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。

このストアド・プロシージャは、db2gse.gse_enable_sref を置き換えます。

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):

```

▶▶ db2gse.ST_create_srs(—srs_name—, —srs_id—, —x_offset—, —x_scale—
                        [NULL],
▶, —y_offset—, —y_scale—, —z_offset—, —z_scale—,
  [NULL] [NULL] [NULL] [NULL],
▶ —m_offset—, —m_scale—, —coordsys_name—, —description—)
  [NULL] [NULL] [NULL]

```

可能な最大範囲を使用する場合 (バージョン 2):

```

▶▶ db2gse.ST_create_srs(—srs_name—, —srs_id—, —x_min—, —x_max—,
▶ —x_scale—, —, —y_min—, —y_max—, —y_scale—, —z_min—, —z_max—,
  [NULL]
▶ —z_scale—, —m_min—, —m_max—, —m_scale—, —coordsys_name—,
  [NULL]
▶ —description—)
  [NULL]

```

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 まででなければなりません。DB2 Geodetic Extender 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図形の、すべての X 座標のオフセットを指定します。図形が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール因数 *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図形の、すべての X 座標のスケール因数を指定します。図形が外部表記 (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、スケール因数が適用されます (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図形の、すべての Y 座標のオフセットを指定します。図形が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール因数 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図形の、すべての Y 座標のスケール因数を指定します。図形が外部表記 (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、スケール因数が適用されます (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図形の、すべての Z 座標のオフセットを指定します。図形が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表記に変換される時に、スケール因数 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図形の、すべての Z 座標のスケール因数を指定します。図形が外部表記 (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、スケール因数が適用されます (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図形の、すべての M 座標のオフセットを指定します。図形が外部表記 (WKT、WKB、形状) から DB2 Spatial Extender の内部表

記に変換される時に、スケール因数 *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図形の、すべての M 座標のスケール因数を指定します。図形が外部表記 (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、スケール因数が適用されます (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系を一意的に識別します。座標系は、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せん。

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図形の、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図形の、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図形の、すべての X 座標のスケール因数を指定します。図形が外部表記 (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、スケール因数が適用されます (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図形の、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図形の、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図形の、すべての Y 座標のスケール因数を指定します。図形が外部表記 (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、スケール因数が適用されます (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図形の、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図形の、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図形の、すべての Z 座標のスケール因数を指定します。図形が外部表記 (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、スケール因数が適用されます (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図形の、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図形の、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図形の、すべての M 座標のスケール因数を指定します。図形が外部表記 (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、スケール因数が適用されます (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系を一意的に識別します。座標系は、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

プロシージャから戻される、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を使用して、ST_create_srs ストアード・プロシージャ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・プロシージャの実行後に表示されます。

関連概念:

- 70 ページの『空間参照系』

関連タスク:

- 77 ページの『空間参照系の作成』

ST_disable_autogeocoding

このストアド・プロシージャは、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。

このストアド・プロシージャは、db2gse.gse_disable_autogc を置き換えます。

許可:

このストアド・プロシージャ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、SYSADM 権限または DBADM 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:

```
▶ db2gse.ST_disable_autogeocoding ( ( table_schema , table_name , column_name ) )
```

NULL

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。

ST_disable_autogeocoding

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

ストアド・プロシージャ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

プロシージャから戻される、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を使用して、ST_disable_autogeocoding ストアド・プロシージャ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、ストアド・プロシージャの実行後に表示されます。

関連資料:

- 262 ページの『ST_enable_autogeocoding』
- 289 ページの『ST_setup_geocoding』

ST_disable_db

このストアド・プロシージャは、DB2 Spatial Extender が、空間データを保管し、このデータに対して行われる操作をサポートするために必要なリソースを除去するために使用します。

このストアド・プロシージャは、データベースを地理情報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。たとえば、データベースを地理情報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、このストアド・プロシージャを呼び出して、最初のデータベースからすべての地理情報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、これらの型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地理情報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地理情報リソース (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。

このストアード・プロシージャは db2gse.gse_disable_db を置き換えます。

許可:

このストアード・プロシージャ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、SYSADM または DBADM 権限のいずれかを持つ必要があります。

構文:

```
▶▶ db2gse.ST_disable_db ( ( force ) )
```

└───┬───┘
NULL

パラメーターの説明:

force

地理情報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、データベースを地理情報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・オブジェクトが地理情報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、およびその他のデータ・タイプ (地理情報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

プロシージャから戻される、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:

ST_disable_db

この例は、DB2 コマンド行プロセッサを使用して、ST_disable_db ストアド・プロシージャ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、データベースの地理情報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、ストアド・プロシージャの実行後に表示されます。

関連資料:

- 242 ページの『ST_alter_coordsys』
- 248 ページの『ST_create_coordsys』

ST_drop_coordsys

このストアド・プロシージャは、データベースから座標系についての情報を削除するために使用します。このストアド・プロシージャが処理されると、座標系についての情報が 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系をドロップすることはできません。

許可:

このストアド・プロシージャを呼び出すユーザー ID は、SYSADM または DBADM 権限のいずれかを持つ必要があります。

構文:

```
▶▶—db2gse.ST_drop_coordsys—(—coordsys_name—)—————▶▶
```

パラメーターの説明:

coordsys_name

座標系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

ストアド・プロシージャ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

プロシージャから戻される、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を使用して、ST_drop_coordsys ストアド・プロシージャ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 という名前の座標系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、ストアド・プロシージャの実行後に表示されます。

ST_drop_srs

このストアド・プロシージャは、空間参照系をドロップするために使用します。このストアド・プロシージャが処理されると、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地理情報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: このストアド・プロシージャを使用する場合は注意が必要です。このストアド・プロシージャ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地理情報操作は実行できなくなります。

このストアド・プロシージャは db2gse.gse_disable_sref を置き換えます。

許可:

このストアド・プロシージャを呼び出すユーザー ID は、SYSADM または DBADM 権限のいずれかを持つ必要があります。

構文:

```
►►db2gse.ST_drop_srs(—srs_name—)◄◄
```

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。

ST_drop_srs

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

プロシージャから戻される、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を使用して、ST_drop_srs ストアード・プロシージャ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・プロシージャの実行後に表示されます。

関連資料:

- 250 ページの『ST_create_srs』
- 244 ページの『ST_alter_srs』

ST_enable_autogeocoding

このストアード・プロシージャは、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、トリガーが活動化されます。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。

このストアード・プロシージャは `db2gse.gse_enable_autogc` を置き換えます。

許可:

このストアード・プロシージャ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、SYSADM 権限または 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 SYSADM または 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
- ジョコーディング・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:

```
▶—db2gse.ST_enable_autogeocoding—(—table_schema—,—table_name—,—————▶
                                   |————|
                                   |NULL|
▶—column_name—)————▶
```

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。

ST_enable_autogeocoding

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

プロシージャから戻される、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を使用して、ST_enable_autogeocoding ストアード・プロシージャ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・プロシージャの実行後に表示されます。

関連資料:

- 289 ページの『ST_setup_geocoding』

ST_enable_db

このストアード・プロシージャは、空間データを保管し、地理情報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、地理情報索引タイプ、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・プロシージャが含まれます。

このストアード・プロシージャは、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 SYSADM または DBADM 権限のいずれかを持つ必要があります。

構文:

```
▶▶ db2gse.ST_enable_db ( ( table_creation_parameters ) )
```

└───┬───┘
NULL

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。たとえば、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

プロシージャから戻される、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;
```

ST_enable_db

```
/* 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);
```

関連資料:

- 258 ページの『ST_disable_db』

ST_export_shape

このストアード・プロシージャは、地理情報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。

このストアード・プロシージャは db2gse.gse_export_shape を置き換えます。

許可:

このストアード・プロシージャ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:

```

▶▶ db2gse.ST_export_shape (—file_name—, —append_flag—, —————▶
                             |NULL|
▶ —output_column_names—, —select_statement—, —messages_file—) —————▶
   |NULL|                                     |NULL|

```

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 ストアド・プロシージャを使用して、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すでに存在する場合、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、269 ページの『使用上の注意』を参照してください。ストアド・プロシージャは、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、地理情報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、地理情報列を除き、*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 つの地理情報列および、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 (たとえば、座標系が異なるなどの理由から)

ストアド・プロシージャは、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*

ストアド・プロシージャ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

プロシージャから戻される、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:

1 度にエクスポートできる地理情報列は 1 つだけです。

ST_export_shape ストアード・プロシージャは次の 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)。
- 地理情報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)。座標系は、最初の地理情報レコードから得られます。後続のレコードが異なる座標系を持つ場合、エラーが起こります。

次の表は、DB2 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 データ・タイプはすべて、サポートされません。

表 29. 属性ファイル内での DB2 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を使用して、ST_export_shape ストアード・プロシージャ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',?,?)
```

ST_export_shape

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・プロシージャの実行後に表示されます。

関連資料:

- 270 ページの『ST_import_shape』

ST_import_shape

このストアード・プロシージャは、地理情報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 つの地理情報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地理情報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。

このストアード・プロシージャは `db2gse.gse_import_shape` を置き換えます。

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。
 - SYSADM または DBADM
 - 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。
 - SYSADM または 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:

```
▶▶ db2gse.ST_import_shape (—file_name—, —input_attr_columns—, —————→  
                           └─NULL─┘
```



```

▶-srs_name-, [table_schema NULL], -table_name-, [table_attr_columns NULL],
▶[create_table_flag NULL], [table_creation_parameters NULL], -spatial_column-
▶-, [type_schema NULL], [type_name NULL], [inline_length NULL], [id_column NULL]
▶-, [id_column_is_identity NULL], [restart_count NULL], [commit_scope NULL],
▶[exception_file NULL], [messages_file NULL])

```

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・マシンに存在しなければならない、必須ファイルのリストについては、277 ページの『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列名はコンマで区切る必要があります。結果の列名は、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

地理情報列にインポートされる図形に使用される、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。

地理情報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) に指定された座標系と SRS の座標系とを比較します。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確認します。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確認します。

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 データ・タイプとどのように対応付けられるかについては、277 ページの『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。(表がすでに存在する場合は、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、表はすで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。たとえば、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

形状ファイルのロード先の表の、地理情報列の名前。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、作成される新しい地理情報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地理情報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

新しい表に地理情報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

地理情報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・タイプは地理情報列のデータ・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、表内の地理情報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地理情報レコードは 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。

各種の地理情報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図形の数および、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、表がすで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 文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。たとえば、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て、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 (たとえば、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せん。

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、277 ページの『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。ファイルがすでに存在する場合、ストアード・プロシージャ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 (たとえば、座標系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。ファイルがすでに存在する場合、ストアード・プロシージャ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

プロシージャから戻される、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 ストアド・プロシージャは、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。
- 形状データの座標系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系が、*srs_id* パラメーターで指定された空間参照系の座標系と比較されます。

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 30. 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 10 進 b (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	<i>len</i>	<i>dec</i>	DECIMAL(<i>len,dec</i>)	<i>len</i> <32
F	<i>len</i>	<i>dec</i>	REAL	<i>len</i> + <i>dec</i> < 7
F	<i>len</i>	<i>dec</i>	DOUBLE	
C	<i>len</i>		CHAR(<i>len</i>)	
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。

ST_import_shape

たとえば、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を使用して、ST_import_shape ストアード・プロシージャ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・プロシージャの実行後に表示されます。

関連資料:

- 266 ページの『ST_export_shape』

ST_register_geocoder

このストアード・プロシージャは、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すで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
 - 図形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。たとえば、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図形を作成する必要があります。図形は、既存の空間参照系を使用して表現する必要があります。図形に対して ST_IsValid 空間処理関数を呼び出した時に値 1 が戻されれば、その図形は有効です。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。

ジオコーダーがすで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。

このストアード・プロシージャは db2gse.gse_register_gc を置き換えます。

許可:

このストアード・プロシージャを呼び出すユーザー ID は、このストアード・プロシージャが登録するジオコーダーを含むデータベースに対して、SYSADM または DBADM 権限のいずれかを持つ必要があります。

構文:

```
▶▶ db2gse.ST_register_geocoder ( ( geocoder_name , function_schema ,
▶▶ function_name , specific_name , default_parameter_values ,
▶▶ parameter_descriptions , vendor , description ) )
```

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。関数はすでに作成済みで、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。関数はすでに作成済みで、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。たとえば、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。たとえば、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。たとえば、以下のよう指定します。

```
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、座標系の名前
- 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

ST_register_geocoder

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

ストアド・プロシージャ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

プロシージャから戻される、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 `CALL` コマンドを使用して、`ST_register_geocoder` ストアド・プロシージャを呼び出し、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、ストアド・プロシージャの実行後に表示されます。

関連資料:

- 293 ページの『ST_unregister_geocoder』

ST_register_spatial_column

このストアド・プロシージャは、地理情報列を登録し、この列と空間参照系 (SRS) を関連付けるために使用します。このストアド・プロシージャが処理されると、登録される地理情報列の情報が DB2GSE.ST_GEOMETRY_COLUMNS カタログ・ビューに追加されます。地理情報列を登録すると、可能な場合、すべての図形が指定された SRS を必ず使用するよう、表に制約が作成されます。

このストアド・プロシージャは db2gse.gse_register_layer を置き換えます。

許可:

このストアド・プロシージャ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。

- 登録される地理情報列が属する表を含むデータベースに対する、SYSADM 権限または DBADM 権限。
- この表に対する CONTROL または ALTER 特権。

構文:

```
► db2gse.ST_register_spatial_column( ( table_schema , table_name ,
  └───┬───┘
    NULL
) , column_name , srs_name )
```

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。

ST_register_spatial_column

srs_name

この地理情報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

プロシージャから戻される、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を使用して、ST_register_spatial_column ストアード・プロシージャ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 という名前の地理情報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・プロシージャの実行後に表示されます。

関連資料:

- 294 ページの『ST_unregister_spatial_column』

ST_remove_geocoding_setup

このストアード・プロシージャは、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:

このストアド・プロシージャ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、SYSADM 権限または DBADM 権限。
- この表に対する CONTROL 特権または UPDATE 特権。

構文:

```
▶ db2gse.ST_remove_geocoding_setup—(—table_schema—, —table_name—, —
      |_____
      |NULL_____
      |_____
▶ —column_name—)—————▶
```

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

ストアド・プロシージャ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

ST_remove_geocoding_setup

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

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、ストアド・プロシージャーの実行後に表示されます。

関連資料:

- 289 ページの『ST_setup_geocoding』

ST_run_geocoding

このストアド・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。

このストアド・プロシージャーは db2gse.gse_run_gc を置き換えます。

許可:

このストアド・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、SYSADM 権限または DBADM 権限。
- この表に対する CONTROL 特権または UPDATE 特権。

構文:

```
▶▶ db2gse.ST_run_geocoding—(—table_schema—, —table_name—, —————▶  
                                  NULL                                  NULL  
▶—column_name—, —geocoder_name—, —parameter_values—, —————▶  
                                  NULL                                  NULL  
▶—where_clause—, —commit_scope—) —————▶▶  
                                  NULL                                  NULL
```

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。たとえば、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。たとえば、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 文節の本体を指定し、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* 文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文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

プロシージャから戻される、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を使用して、ST_run_geocoding ストアード・プロシージャ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・プロシージャの実行後に表示されます。

関連資料:

- 289 ページの『ST_setup_geocoding』

ST_setup_geocoding

このストアード・プロシージャは、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。

ST_setup_geocoding

このストアード・プロシージャ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、ランタイム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、SYSADM 権限または DBADM 権限。
- この表に対する CONTROL 特権または UPDATE 特権。

構文:

```
▶▶ db2gse.ST_setup_geocoding—( ( table_schema , table_name ,  
                                └───┬───┘  
                                NULL  
▶ column_name , geocoder_name , ( parameter_values ,  
                                   └───┬───┘  
                                   NULL  
▶ ( autogeocoding_columns , where_clause , commit_scope ) )▶▶  
   └───┬───┘ └───┬───┘ └───┬───┘  
   NULL      NULL      NULL
```

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。たとえば、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。たとえば、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、トリガーが活動化されます。

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 文節の本体を指定し、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 文節に制限は定義されません。

文節は、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

プロシージャから戻される、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を使用して、ST_setup_geocoding ストアド・プロシージャ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 という名前の地理情報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,,,$HOME/sql1lib/
    gse/refdata/ky.edg','$HOME/sql1lib/samples/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、ストアド・プロシージャの実行後に表示されます。

関連資料:

- 293 ページの『ST_unregister_geocoder』
- 284 ページの『ST_remove_geocoding_setup』

ST_unregister_geocoder

このストアド・プロシージャは、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。

このストアド・プロシージャは db2gse.gse_unregister_gc を置き換えます。

許可:

このストアド・プロシージャ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、SYSADM または DBADM 権限のいずれかを持つ必要があります。

構文:

```
►►—db2gse.ST_unregister_geocoder—(—geocoder_name—)—————►►
```

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。

ST_unregister_geocoder

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

ストアド・プロシージャ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

プロシージャから戻される、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を使用して、ST_unregister_geocoder ストアド・プロシージャ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、ストアド・プロシージャの実行後に表示されます。

関連資料:

- 278 ページの『ST_register_geocoder』
- 289 ページの『ST_setup_geocoding』

ST_unregister_spatial_column

このストアド・プロシージャは、地理情報列の登録を抹消するために使用します。このストアド・プロシージャは、次のようにして登録を抹消します。

- 空間参照系と地理情報列の関連付けを除去する。ST_GEOMETRY_COLUMNS カタログ・ビューにはまだ地理情報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 (この地理情報列内の図形値が、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。

このストアド・プロシージャは db2gse.gse_unregister_layer を置き換えます。

許可:

このストアード・プロシージャ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。

- SYSADM または DBADM 権限
- この表に対する CONTROL または ALTER 特権。

構文:

```
▶▶db2gse.ST_unregister_spatial_column—(—table_schema—,—table_name—,—  
NULL—  
—column_name—)
```

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

登録を抹消する地理情報列の名前。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

プロシージャから戻される、メッセージ・コードに付随する実際のメッセー

ST_unregister_spatial_column

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を使用して、ST_unregister_spatial_column ストアド・プロシージャ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 という名前の地理情報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、ストアド・プロシージャの実行後に表示されます。

関連資料:

- 283 ページの『ST_register_spatial_column』

第 21 章 カタログ・ビュー

Spatial Extender のカタログ・ビューには、次の情報が含まれています。

『DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビュー』

使用する座標系

298 ページの『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

301 ページの『DB2GSE.ST_GEOCODERS カタログ・ビュー』 および 303 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

301 ページの『DB2GSE.ST_GEOCODING カタログ・ビュー』 および 303 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

304 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

305 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

307 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_COORDINATE_SYSTEMS カタログ・ビュー

登録済みの座標系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系を Spatial Extender カタログに自動的に登録します。

- 地理情報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系を定義します。

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?	説明
COORDSYS_NAME	VARCHAR(128)	いいえ	この座標系の名前。名前はデータベース内でユニークなもの。

DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビュー

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?	説明
COORDSYS_TYPE	VARCHAR(128)	いいえ	この座標系のタイプ PROJECTED 2 デイメンション。 GEOGRAPHIC 3 デイメンション。X 座標と Y 座標を使用。 GEOCENTRIC 3 デイメンション。X、Y、Z の座標を使用。 UNSPECIFIED 抽象または非現実世界の座標系。 この列の値は DEFINITION 列から得られる。
DEFINITION	VARCHAR(2048)	いいえ	この座標系の定義の既知テキスト表記
ORGANIZATION	VARCHAR(128)	はい	この座標系を定義した組織 (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。
ORGANIZATION_ COORDSYS_ID	INTEGER	はい	座標系を定義した組織によってこの座標系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系が識別される。 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系の説明

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。地理情報列が、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・ビューを調べます。

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?	説明
TABLE_SCHEMA	VARCHAR(128)	いいえ	この地理情報列が入った表が属するスキーマの名前。
TABLE_NAME	VARCHAR(128)	いいえ	この地理情報列が入った表の非修飾名。

表 32. 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?	説明
COLUMN_NAME	VARCHAR(128)	いいえ	この地理情報列の名前。 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	この地理情報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	この地理情報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	この地理情報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	この地理情報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

データベースで地理情報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登録されます。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?	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。

DB2GSE.ST_GEOCODER_PARAMETERS カタログ・ビュー

表 33. DB2GSE.ST_GEOCODER_PARAMETERS の列 (続き)

名前	データ・タイプ	NULL 可能か?	説明
ORDINAL	SMALLINT	いいえ	<p>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。</p> <p>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。</p> <p>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。</p>
PARAMETER_NAME	VARCHAR(128)	はい	<p>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。</p> <p>PARAMETER_NAME 列の内容は、DB2 カタログから得られる。</p>
TYPE_SCHEMA	VARCHAR(128)	いいえ	<p>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。</p>
TYPE_NAME	VARCHAR(128)	いいえ	<p>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。</p>
PARAMETER_DEFAULT	VARCHAR(2048)	はい	<p>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。</p> <p>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。</p>
説明	VARCHAR(256)	はい	<p>アプリケーションを示すパラメーターの説明。</p>

DB2GSE.ST_GEOCODERS カタログ・ビュー

地理情報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に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 Spatial Extender の DB2GSE.ST_GEOCODER_PARAMETERS カタログ・ビューと DB2 の SYSCAT.ROUTINEPARMS カタログ・ビューを参照してください。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・ビューを参照してください。

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?	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。
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・ビューを参照してください。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。たとえば、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 カタログ・ビュー

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。たとえば、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?	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地理情報列の名前。 TABLE_SCHEMA、TABLE_NAME、COLUMN_NAME の列の結合値によって、地理情報列が一意的に識別される。
GEOCODER_NAME	VARCHAR(128)	いいえ	COLUMN_NAME 列で指定される地理情報列にデータを生成するジオコーダーの名前。1 つの地理情報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、活動化されている。 INVALID 地理情報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 文節内の検索条件。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ド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。たとえば、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 と呼ばれるこの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・ビューを参照してください。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?	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	地理情報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地理情報列の名前。 TABLE_SCHEMA、TABLE_NAME、 COLUMN_NAME の列の結合値によって、地理情報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。

DB2GSE.ST_GEOCODING_ PARAMETERS カタログ・ビュー

表 36. DB2GSE.ST_GEOCODING_PARAMETERS カタログ・ビューの列 (続き)

名前	データ・タイプ	NULL 可能か?	説明
PARAMETER_VALUE	VARCHAR(2048)	はい	<p>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。</p> <p>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。</p>

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。たとえば、*coordinate system name*、*geocoder name*、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 (たとえば、座標系名、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?	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。

表 37. DB2GSE.ST_SIZINGS カタログ・ビューの列 (続き)

名前	データ・タイプ	NULL 可能か?	説明
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 列の可能な値は、以下のとおり。 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。
説明	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・ビューを参照してください。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。詳細については、71 ページの『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』、および 223 ページの『DB2 Geodetic Extender によってサポートされる測地系』を参照してください。
- 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系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、座標系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系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 列を参照してください。座標系の詳細については、DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビューを参照してください。

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?	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。

DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビュー

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?	説明
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	図形のすべての X 座標から減算されるオフセット。減算は、図形座標を、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因数が乗算される。
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乘算されるスケール因数。この因数は Y_SCALE 列に示された値と同一である。
Y_OFFSET	DOUBLE	いいえ	図形のすべての Y 座標から減算されるオフセット。減算は、図形座標を、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因数が乗算される。
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乘算されるスケール因数。この因数は X_SCALE 列に示された値と同一である。
Z_OFFSET	DOUBLE	いいえ	図形のすべての Z 座標から減算されるオフセット。減算は、図形座標を、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因数が乗算される。
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乘算されるスケール因数。
M_OFFSET	DOUBLE	いいえ	図形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因数が乗算される。
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乘算されるスケール因数。
MIN_X	DOUBLE	いいえ	この空間参照系が適用される、図形の 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。
MAX_X	DOUBLE	いいえ	この空間参照系が適用される、図形の 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、図形の 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。

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?	説明
MAX_Y	DOUBLE	いいえ	この空間参照系が適用される、図形の 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、図形の 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、図形の 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、図形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、図形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系の識別名。
COORDSYS_TYPE	VARCHAR(128)	いいえ	この空間参照系の基礎となる座標系のタイプ。
ORGANIZATION	VARCHAR(128)	はい	この空間参照系の基礎となる座標系を定義した組織名 (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系を定義した組織名 (標準化団体など)。 ORGANIZATION が NULL の場合は ORGANIZATION_COORSYS_ID は NULL である。
DEFINITION	VARCHAR(2048)	いいえ	座標系の定義の事前割り当てテキスト表記
説明	VARCHAR(256)	はい	空間参照系の説明。

関連概念:

- 70 ページの『空間参照系』

関連タスク:

- 77 ページの『空間参照系の作成』

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。たとえば、ST_Distance は、2 つの指定した図形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。

DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビュー

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?	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。可能な値は、以下のとおり。 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。
説明	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

DB2® Spatial Extender は、次のことを行う関数を提供します。

- 図形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図形を比較する。これらの関数は、**比較関数**と呼ばれています。
- 図形内の座標や指標、図形間の関係、および境界やその他の情報など、図形のプロパティに関する情報を戻す。
- 既存の図形から新規の図形を生成する。
- 図形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。
- 異なる座標系の間で投影や変換を行う。

関連概念:

- 346 ページの『距離情報を戻す関数』
- 347 ページの『索引情報を戻す関数』
- 347 ページの『座標系間の変換』

関連資料:

- 127 ページの『空間処理関数による操作の例』
- 331 ページの『図形のプロパティについての情報を戻す関数』
- 318 ページの『地勢を比較する関数』
- 338 ページの『既存の図形から新規図形を生成する関数』
- 309 ページの『図形値をデータ交換フォーマットに変換する空間処理関数』

図形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、図形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
- ジオグラフィー・マークアップ言語 (GML) 表記

これらのフォーマットから図形を作成する関数は、**コンストラクター関数**として知られています。

関連概念:

- 310 ページの『コンストラクター関数の概要』
- 313 ページの『事前割り当てテキスト (WKT) 表記への変換』
- 315 ページの『事前割り当てバイナリー (WKB) 表記への変換』
- 316 ページの『ESRI 形状表記への変換』
- 317 ページの『Geography Markup Language (GML) 表記への変換』

関連資料:

- 521 ページの『トランスフォーム・グループ』

コンストラクター関数の概要

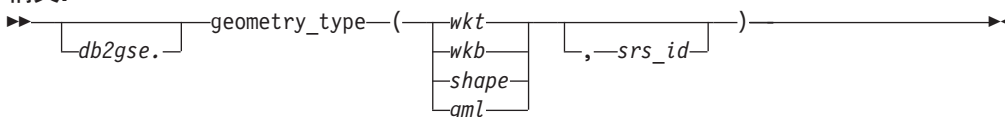
コンストラクター関数は、データが挿入される列の図形データ・タイプと同じ名前をもっています。これらの関数は、常にそれぞれの入力データ交換フォーマットに関して働きます。このセクションでは、以下について述べます。

- 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図形のタイプ
- 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図形のタイプ
- コードおよび結果セットの例

データ交換フォーマットに関して働く関数

このセクションでは、データ交換フォーマットに関して働く呼び出し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図形のタイプを明らかにします。

構文:



パラメーターおよび構文のその他のエレメント:

db2gse DB2[®] Spatial Extender によって提供される空間データ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* 図形の事前割り当てテキスト表記が入る、タイプが CLOB(2G) の値。
- wkb* 図形の事前割り当てバイナリー表記が入る、タイプが BLOB(2G) の値。
- shape* 図形の ESRI 形状表記が入る、タイプが BLOB(2G) の値。
- gml* 図形の Geography Markup Language (GML) 表記が入る、タイプが CLOB(2G) の値。
- srs_id* 結果の図形の空間参照系を識別する、タイプが 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。

戻りタイプ:

geometry_type

geometry_type が *ST_Geometry* の場合、戻された図形タイプの動的タイプは、入力値によって示された図形に対応します。

geometry_type がそれ以外のタイプの場合、戻された図形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図形が、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。

座標から図形を作成する関数

ST_Point 関数は、データ交換フォーマットからだけでなく、数値座標値からも図形を作成します。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。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、およびそれが戻す図形のタイプについての情報を記載しています。

構文:

```
db2gse.ST_Point(—coordinates |—————)—————▶
                |—————|
                |,—srs_id—|
```

coordinates:

```
|—x_coordinate—,—y_coordinate—|—————|
                |—————|
                |,—z_coordinate—|—————|
                |—————|
                |,—m_coordinate—|
```

パラメーター:

x_coordinate

結果のポイントの X 座標を示す、タイプが DOUBLE の値。

y_coordinate

結果のポイントの Y 座標を示す、タイプが DOUBLE の値。

z_coordinate

結果のポイントの Z 座標を示す、タイプが DOUBLE 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、タイプが DOUBLE 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、タイプが 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

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"
```

地理情報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、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) からポイント図形を作成します。2番目の挿入は、数値座標値からポイント図形を作成します。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図形を挿入するために JDBC を使用し、図形を指定するのに 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();
```

関連資料:

- 521 ページの『トランスフォーム・グループ』

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、図形を 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図形値を WKT スtring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、図形をバインドアウトするとき、それら図形を、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、図形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外に、DB2® Spatial Extender は、事前割り当てテキスト表記との間で図形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 standard に準拠するように、これら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_WKTTToSQL**
- **ST_GeomFromText**
- **ST_GeomCollFromText**
- **ST_PointFromText**

- **ST_LineFromText**
- **ST_PolyFromText**
- **ST_MPointFromText**
- **ST_MLineFromText**
- **ST_MPolyFromText**

関連資料:

- 521 ページの『トランスフォーム・グループ』

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図形値を、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、図形をバインドアウトするときに、それら図形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、図形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外に、事前割り当てバイナリー表記との間で図形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 standard に準拠するように、これら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_WKBTtoSQL**
- **ST_GeomFromWKB**
- **ST_GeomCollFromWKB**
- **ST_PointFromWKB**
- **ST_LineFromWKB**
- **ST_PolyFromWKB**
- **ST_MPointFromWKB**
- **ST_MLineFromWKB**
- **ST_MPolyFromWKB**

関連資料:

- 521 ページの『トランスフォーム・グループ』

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図形値を、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、図形をバインドアウトするときに、それら図形を、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、図形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。

関連資料:

- 521 ページの『トランスフォーム・グループ』

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、図形を 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図形値を 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、図形をバインドアウトするときに、それら図形を、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、図形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。

関連資料:

- 521 ページの『トランスフォーム・グループ』

地勢を比較する関数

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、図形間、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。これらの情報を提供する関数は、**比較関数** と呼ばれています。

比較関数には以下のものがあります。

- **ST_Contains** および **ST_Within**。これらの関数は両方とも、2 つの図形を入力とし、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。
- **ST_Intersects**、**ST_Crosses**、**ST_Overlaps**、および **ST_Touches**。これらの関数は図形の交差についての情報を戻します。
- **ST_EnvIntersects** および **ST_MBRIntersects**。これらの関数は、1 つの図形を囲む最小の長方形が、もう 1 つの図形を囲む最小の長方形と交差するかどうかを判別します。
- **ST_Equals**、**ST_EqualCoordsys** および **ST_EqualSR**。これらの関数は、比較中の 2 つの図形が同一かどうかを判別します。
- **ST_Relate**。この関数は、比較中の図形が 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_Disjoint**。この関数は、2 つの図形が交差しているかどうかをチェックします。

関連概念:

- 330 ページの『図形を DE-9IM パターン・マトリックス・ストリングと比較する関数』
- 318 ページの『比較関数の概要』
- 320 ページの『ある図形が別の図形を含むかどうかをチェックする関数』
- 323 ページの『図形どうしが交差するかどうかをチェックする関数』
- 328 ページの『図形のエンベロープを比較する関数』
- 328 ページの『2 つのものが同一かどうかをチェックする関数』
- 330 ページの『2 つの図形が交差していないかを確認する関数』

比較関数の概要

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図形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図形間で、ペアワイズ (2 つ一組の) 地理情報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、すべてのタイプの図形間の地理情報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。

図形 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。そして、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) の、I(b)、B(b)、および E(b) との交差が 3 x 3 のマトリックスを構成します。それぞれの交差は、異なるディメンションの図形になります。たとえば、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返されます。

	内部	境界	外部
内部	dim(I(a) \cap I(b))	dim(I(a) \cap B(b))	dim(I(a) \cap E(b))
境界	dim(B(a) \cap I(b))	dim(B(a) \cap B(b))	dim(B(a) \cap E(b))
外部	dim(E(a) \cap I(b))	dim(E(a) \cap B(b))	dim(E(a) \cap E(b))

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。考えられるパターン値は、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ない、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ない、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ない、その最大ディメンションは 2 でなければならない。dim = 2。

たとえば、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 40. ST_Within のマトリックス： 図形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

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 内部	T	*	F
図形 a 境界	*	*	F
図形 a 外部	*	*	*

ST_Within 関数は、両方の図形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、中には色々な図形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 199* で発表されました。

関数のリスト

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

ある図形が別の図形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 つの図形を入力とし、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 番目の図形が 2 番目の図形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 番目の図形が完全に 2 番目の図形内にあるかどうか (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

ST_Contains は、2 番目の図形が 1 番目の図形に完全に包含されていれば値 1 を返します。ST_Contains 関数は、ST_Within 関数と全く正反対の結果を返します。

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。

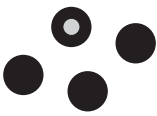
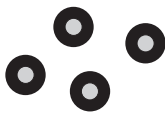
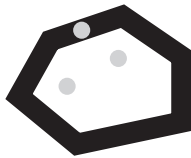



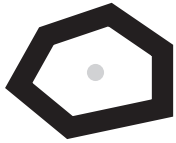


		
複数ポイント / ポイント	複数ポイント / 複数ポイント	ポリゴン / 複数ポイント
		
折れ線 / ポイント	折れ線 / 複数ポイント	折れ線 / 折れ線
		
ポリゴン / ポイント	ポリゴン / 折れ線	ポリゴン / ポリゴン

図 44. ST_Contains : 黒い図形は図形 a を表し、グレーの図形は図形 b を表します。すべてのケースにおいて、図形 a は図形 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、両方の図形の内部は交差しなければならないこと、および 2 番目 (図形 b) の内部または境界は 1 番目 (図形 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、図形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 41. ST_Contains のマトリックス

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 内部	T	*	*
図形 a 境界	*	*	*
図形 a 外部	F	F	*

ST_Within

ST_Within は、1 番目の図形が完全に 2 番目の図形内にある場合、値 1 を返します。ST_Within は、ST_Contains と正反対の結果を返します。

ポイント / 複数ポイント	複数ポイント / 複数ポイント	複数ポイント / ポリゴン
ポイント / 折れ線	複数ポイント / 折れ線	折れ線 / 折れ線
ポイント / ポリゴン	折れ線 / ポリゴン	ポリゴン / ポリゴン

図 45. ST_Within

ST_Within 関数のパターン・マトリックスは、両方の図形の内部が交差しなければならないこと、および 1 次 (図形 a) の内部または境界は 2 次 (図形 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 42. ST_Within のマトリックス

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 内部	T	*	F
図形 a 境界	*	*	F
図形 a 外部	*	*	*

322 ページの図 45 は、ST_Within の適用例を示しています。

- ポイントは、内部が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。
- 322 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。

図形どうし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 つの図形が他の図形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 つの図形が以下に述べる 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。図形の内部が交差すること。図形の境界が交差すること。1 番目の図形の境界が 2 番目の図形の内部と交差すること。または、1 番目の図形の内部が 2 番目の図形の境界と交差すること。
- ST_Crosses は、異なるディメンションの図形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、交差の場所自体が図形と見なされます。ST_Crosses の場合、この図形が、交差する図形より小さいディメンションの図形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図形である必要があります)。たとえば、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 つの図形が交差し、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 に与えられる図形は同じディメンションの図形でなければなりません。ST_Overlaps の場合、これらの図形が一部分オーバーラップし、それらの図形と同じディメンションである新規の図形 (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 つの図形の境界が交差するかどうかを判別します。

ST_Intersects

ST_Intersects は、交差の結果が空の集合にならない場合、値 1 を返します。

ST_Intersects は、ST_Disjoint と正反対の結果を返します。

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 を返す場合、1 を返します。

表 43. ST_Intersects のマトリックス (1)： ST_Intersects 関数は、両方の図形の内部が交差する場合、1 を返します。

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 境界	*	*	*
図形 a 内部	T	*	*
図形 a 外部	*	*	*

表 44. ST_Intersects のマトリックス (2)： ST_Intersects 関数は、1 番目の図形の境界が 2 番目の図形の境界に交差する場合、1 を返します。

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 境界	*	*	*
図形 a 内部	*	T	*
図形 a 外部	*	*	*

表 45. ST_Intersects のマトリックス (3)： ST_Intersects 関数は、1 番目の図形の境界が 2 番目の図形の内部に交差する場合、1 を返します。

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 境界	T	*	*
図形 a 内部	*	*	*
図形 a 外部	*	*	*

表 46. ST_Intersects のマトリックス (4)： ST_Intersects 関数は、いずれかの図形の境界が交差する場合、1 を返します。

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 境界	*	T	*
図形 a 内部	*	*	*
図形 a 外部	*	*	*

ST_Crosses

ST_Crosses は 2 つの図形を入力とし、以下の場合に値 1 を返します。

- 交差が、結果としてディメンションが元の図形の最大ディメンションよりも小さくなるような図形になる場合。
- 交差の集合が、元の両方の図形の内側にある場合。

ST_Crosses は、1 番目の図形が面または複数面である場合、あるいは 2 番目の図形がポイントまたは複数ポイントである場合、NULL を返します。他のすべての組み合わせについては、ST_Crosses は値 1 (2 つの図形が交差することを示す) または値 0 (2 つの図形は交差しない) のいずれかを返します。

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、図形 *b* が図形 *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、図形 *b* のポリゴンの領域に触れています。

黒の図形は図形 *a* を表し、グレーの図形は図形 *b* を表します。

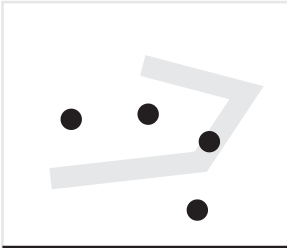
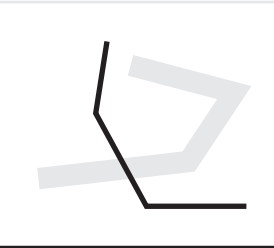

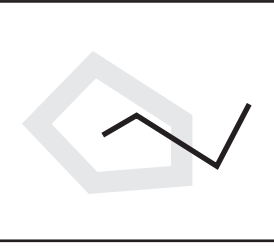
	
複数ポイント / 折れ線	折れ線 / 折れ線
	
複数ポイント / ポリゴン	折れ線 / ポリゴン

図 46. *ST_Crosses*

表 47 のパターン・マトリックスは、1 番目の図形がポイントまたは複数ポイントである場合、あるいは 1 番目の図形が曲線または複数曲線であって、2 番目の図形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 (図形 *a*) の内部が 2 次 (図形 *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 47. *ST_Crosses* のマトリックス (1)

	図形 <i>b</i> 内部	図形 <i>b</i> 境界	図形 <i>b</i> 外部
図形 <i>a</i> 境界	*	*	*
図形 <i>a</i> 内部	T	*	T
図形 <i>a</i> 外部	*	*	*

表 48 のパターン・マトリックスは、1 番目および 2 番目の図形が両方とも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 (図形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 (図形はオーバーラップすることを示す) を戻します。

表 48. *ST_Crosses* のマトリックス (2)

	図形 <i>b</i> 内部	図形 <i>b</i> 境界	図形 <i>b</i> 外部
図形 <i>a</i> 境界	*	*	*
図形 <i>a</i> 内部	0	*	*
図形 <i>a</i> 外部	*	*	*

ST_Overlaps

ST_Overlaps は、同じディメンションの 2 つの図形を比較します。それらの交差の集合が、両者とは異なる図形 (ディメンションは同じ) になる場合、値 1 を返します。

黒の図形は図形 a を表し、グレーの図形は図形 b を表します。すべてのケースにおいて、両方の図形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図形です。このエリアは図形 a および b と同じディメンションをもっています。

以下の図は、図形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。

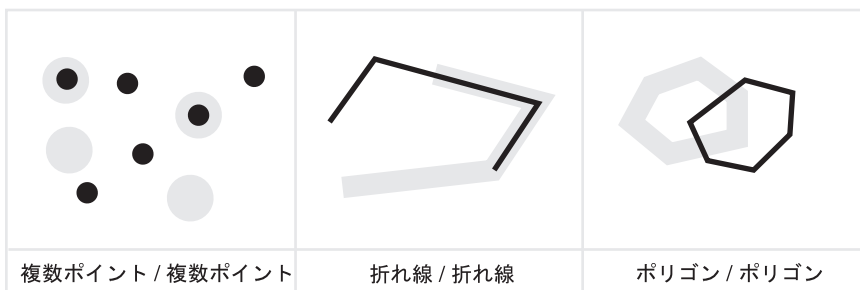


図 47. ST_Overlaps

表 49 のパターン・マトリックスは、1 番目および 2 番目の図形が両方共、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、各図形の内部が他の図形の内部および外部と交差する場合、値 1 を返します。

表 49. ST_Overlaps のマトリックス (1)

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 境界	*	*	*
図形 a 内部	T	*	T
図形 a 外部	T	*	*

表 50 のパターン・マトリックスは、1 番目および 2 番目の図形が両方とも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、図形の交差は、結果としてディメンションが 1 の図形 (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 (図形はオーバーラップしないことを示す) を返します。ただし、ST_Crosses 関数は値 1 (図形は交差することを示す) を返します。

表 50. ST_Overlaps のマトリックス (2)

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 境界	*	*	*
図形 a 内部	1	*	T
図形 a 外部	T	*	*

ST_Touches

ST_Touches は、両方の図形に共通のすべてのポイントが境界上にものみある場合、値 1 を戻します。図形の内部は、互いに交差してはいけません。少なくとも、1 つの図形は曲線、面、複数曲線、または複数面でなければなりません。

黒の図形は図形 *a* を表し、グレーの図形は図形 *b* を表します。すべてのケースにおいて、図形 *b* の境界が図形 *a* と交差しています。図形 *b* の内部は、図形 *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図形が接触している状態を示しています。






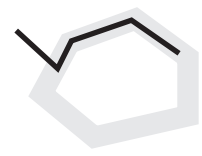
		
ポイント / 折れ線	複数ポイント / 折れ線	折れ線 / 折れ線
		
ポイント / ポリゴン	複数ポイント / ポリゴン	折れ線 / ポリゴン

図 48. ST_Touches

図形の内部が交差せずに、いずれかの図形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 51. ST_Touches のマトリックス (1)

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 境界	*	*	*
図形 a 内部	F	T	*
図形 a 外部	*	*	*

表 52. ST_Touches のマトリックス (2)

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 境界	T	*	*
図形 a 内部	F	*	*
図形 a 外部	*	*	*

表 53. *ST_Touches* のマトリックス (3)

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 境界 *	*	T	*
図形 a 内部 F	F	*	*
図形 a 外部 *	*	*	*

図形のエンベロープを比較する関数

ST_EnvIntersects と *ST_MBRIntersects* は、1 つの図形を囲む最小の長方形が、もう 1 つの図形を囲む最小の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、図形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。

図形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、これら後者の図形が収まる最小のエリアを構成していないからです。これら後者の図形は、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、これらの図形がそれ自体の 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* はそれらの図形自体の交差をテストします。

ST_EnvIntersects

ST_EnvIntersects は、2 つの図形のエンベロープが交差する場合、値 1 を返します。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便宜的な関数です。

ST_MBRIntersects

ST_MBRIntersects は、2 つの図形の最小外接長方形 (MBR) が交差する場合、値 1 を返します。

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 を返します。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 つの図形が同一である場合、値 1 を戻します。図形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、マルチポイント、折れ線、マルチストリング、ポリゴン、マルチポリゴン) で、図形 a と図形 b は同じものです。




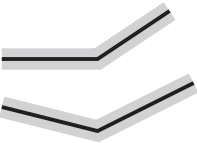
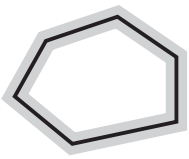
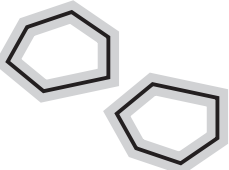
	
ポイント / ポイント	複数ポイント / 複数ポイント
	
折れ線 / 折れ線	複数折れ線 / 複数折れ線
	
ポリゴン / ポリゴン	複数ポリゴン / 複数ポリゴン

図 49. ST_Equals : 黒の図形は図形 a を表し、グレーの図形は図形 b を表します。すべてのケースにおいて、図形 a は図形 b と同じです。

表 54. 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、およびどちらの図形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。

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 境界 *	*	*	F
図形 a 内部 T	T	*	F
図形 a 外部 F	F	F	*

ST_EqualsSRS

ST_EqualsSRS 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 つの図形が交差していないかをチェックする関数

ST_Disjoint は、2 つの図形の交差が空の集合である場合、値 1 を返します。この関数は、ST_Intersects が返すものとは正反対のものを返します。

以下の図は、いずれも、2 つの図形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。

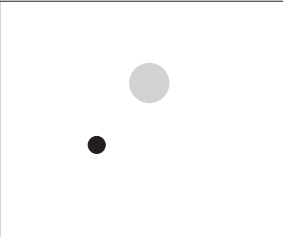
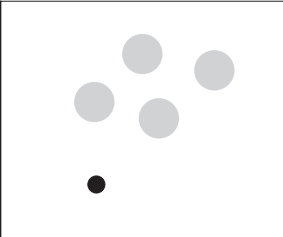
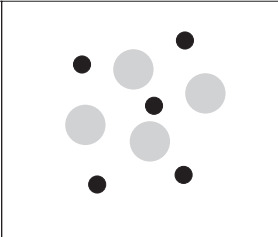
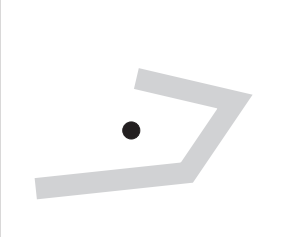




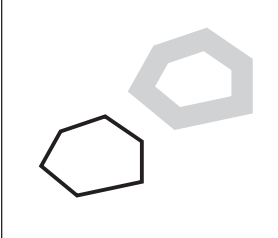
		
ポイント / ポイント	ポイント / 複数ポイント	複数ポイント / 複数ポイント
		
ポイント / 折れ線	複数折れ線 / 折れ線	ポリゴン / 折れ線
		
ポイント / ポリゴン	複数ポイント / 複数ポリゴン	ポリゴン / ポリゴン

図 50. ST_Disjoint : 黒の図形は図形 a を表し、グレーの図形は図形 b を表します。すべてのケースにおいて、図形 a および図形 b は互いに交わりません。

表 55. ST_Disjoint のマトリックス : このマトリックスは、単にどちらの図形の内部も境界も交差しないことを表しています。

	図形 b 内部	図形 b 境界	図形 b 外部
図形 a 境界	F	F	*
図形 a 内部	F	F	*
図形 a 外部	*	*	*

図形を 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 つの図形を比較し、それらの図形が 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 を返します。そうでない場合は、関数は値 0 (ゼロ) を返します。

図形のプロパティについての情報を戻す関数

このセクションでは、図形のプロパティ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。

- 図形のデータ・タイプ
- 図形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)
- デイメンション
- 閉じている、空である、または単純であるという性質
- 図形集合内の基本図形
- 空間参照系

プロパティによっては、それ自体で図形であるものがあります。たとえば、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。これらの図形は、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。他の種類の図形 — たとえば、与えられたロケーションを囲むゾーンを表す図形 — を作成する関数は、別のカテゴリーに属します。『新規図形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。

関連概念:

- 331 ページの『データ・タイプ情報を戻す関数』
- 331 ページの『座標および指標に関する情報を戻す関数』
- 333 ページの『図形内の図形に関する情報を戻す関数』
- 335 ページの『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』
- 336 ページの『図形のデイメンションに関する情報を戻す関数』
- 337 ページの『図形が閉じているか、空か、または単純であるかを示す関数』
- 337 ページの『図形の空間参照系を識別する関数』

データ・タイプ情報を戻す関数

`ST_GeometryType` は図形を入力パラメーターとし、その図形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、図形内の座標および指標に関する情報を戻します。たとえば、`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` は図形内の最高 X 座標を戻し、`ST_MinX` は図形内の最低 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、図形がもっている座標のタイプを表す値、および図形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と同じものではありません。後者は、図形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も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 は図形を入力パラメーターとし、与えられた図形が **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 は図形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。図形は、構造化されたタイプのすべての属性が図形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 は図形を入力パラメーターとし、与えられた図形が **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 は図形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 は図形を入力パラメーターとし、その最大 X 座標を返します。

ST_MaxY

ST_MaxY は図形を入力パラメーターとし、その最大 Y 座標を返します。

ST_MaxZ

ST_MaxZ は図形を入力パラメーターとし、その最大 Z 座標を返します。

ST_MinM

ST_MinM は図形を入力パラメーターとし、その最小指標を返します。

ST_MinX

ST_MinX は図形を入力パラメーターとし、その最小 X 座標を返します。

ST_MinY

ST_MinY は図形を入力パラメーターとし、その最小 Y 座標を返します。

ST_MinZ

ST_MinZ は図形を入力パラメーターとし、その最小 Z 座標を返します。

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 座標を返すことができます。

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 座標を返すことができます。

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 座標を返すことができます。

図形内の図形に関する情報を返す関数

以下の関数は、図形内の図形に関する情報を返します。いくつかの関数は図形内の特定のポイントを識別し、他の関数は集合内の基本図形数を返します。

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 は図形を入力パラメーターとし、図形の中心を戻します。図形の中心とは、与えられた図形の最小の長方形範囲の中心のポイントです。

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、図形の集合と 1 つの索引を入力パラメーターとし、集合の中から、索引で指定された図形を戻します。

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に戻します。

ST_NumGeometries

ST_NumGeometries は図形集合を入力パラメーターとし、その集合内にある図形の数を戻します。

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、図形を入力パラメーターとし、その図形を定義するために使用されたポイントの数を戻します。たとえば、図形がポリゴンであり、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返します。

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返します。

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返します。

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返します。

境界、エンベロップ、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、図形の内部を外部から分割する境界、つまり図形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返します。たとえば、ST_Boundary は、曲線の形で図形の境界を返します。

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_Boundary

ST_Boundary は図形を入力パラメーターとし、その境界を新しい図形として返します。

ST_Envelope

ST_Envelope は図形を入力パラメーターとし、図形の周りのエンベロップを返します。エンベロッ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 つの図形を入力パラメーターとし、2 つの図形のエンベロープが交差する場合は 1 を返します。それ以外の場合、0 (ゼロ) が返されます。

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返します。

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返します。内部リングは、内部図形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 は図形を入力パラメーターとし、その最小外接長方形を返します。

ST_MBRIntersects

ST_MBRIntersects は、2 つの図形の最小外接長方形 (MBR) が交差する場合、値 1 を返します。

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返します。

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) を返します。

図形のディメンションに関する情報を返す関数

以下の関数は、図形のディメンションに関する情報を返します。たとえば、ST_Area は、与えられた図形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、図形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、与えられた図形がカバーするエリアを、指定された測定単位で返します。

ST_Dimension

ST_Dimension は図形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。

図形が閉じているか、空か、または単純であることを示す関数

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)
- 与えられた図形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、このような図形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 は図形を入力パラメーターとし、与えられた図形が空の場合、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 は図形を入力パラメーターとし、与えられた図形が単純な場合、1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

図形の空間参照系を識別する関数

以下の関数は、図形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、図形を変更またはトランスフォームすることなしに、図形の空間参照系を変更することができます。

ST_SrsId (ST_SRID と呼ばれる)

ST_SrsId (または ST_SRID) は、図形および (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、関数は図形の空間参照系を指定した空間参照系に変更した図形が戻されます。図形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、与えられた図形の現行の空間参照系 ID が戻されます。

ST_SrsName

ST_SrsName は図形を入力パラメーターとし、与えられた図形を表現する空間参照系の名前を戻します。

既存の図形から新規図形を生成する関数

このセクションでは、既存の図形から新規図形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、他の図形のプロパティを表す図形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。

- 図形を他の図形に変換する
- スペースの構成を表す図形を作成する
- 複数の図形から個別の図形を派生させる
- 指標に基づいて図形を作成する
- 図形の修正版を作成する

関連概念:

- 338 ページの『ある図形を別の図形に変換する関数』
- 339 ページの『異なる空間構成を使用して新規図形を作成する関数』
- 343 ページの『複数の図形から 1 つの図形を派生させる関数』
- 344 ページの『指標に基づいて新規図形を派生させる関数』
- 345 ページの『既存の図形の修正フォームを作成する関数』

ある図形を別の図形に変換する関数

以下の関数は、スーパー・タイプの図形を、対応するサブタイプの図形に変換することができます。たとえば、ST_ToLineString 関数は、タイプが ST_Geometry 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、基本図形および図形集合を、単一の図形集合に結合することもできます。たとえば、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 は図形を入力パラメーターとし、これを図形の集合に変換します。

ST_ToLineString

ST_ToLineString は図形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 は図形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 は図形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 は図形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 は図形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 は図形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図形を作成する関数

既存の図形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図形を作成します。たとえば、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、既存の図形から指定された半径だけ外側に広がる、新規図形を生成することができます。既存の図形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、新規の図形は面になります。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。

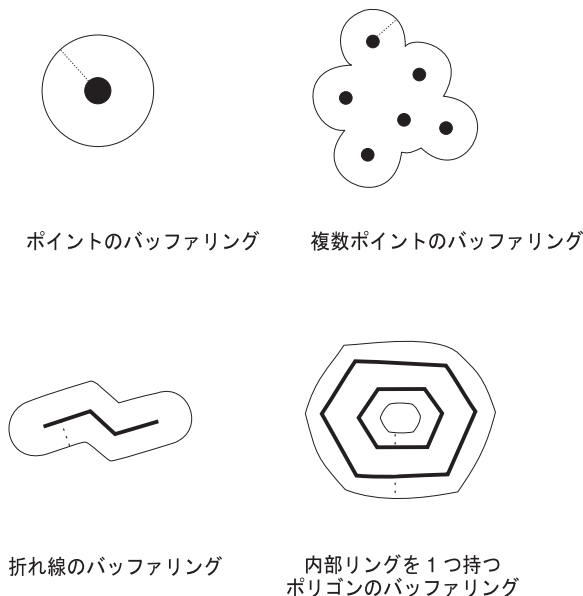


図 51. ST_Buffer

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 の図形 (面および複数面) のみが負のバッファを使用します。元の図形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図形) は常に、バッファ距離の絶対値が使用されます。

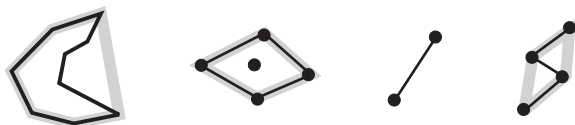
一般的に、外部リングの場合、正のバッファ距離は、元の図形の中心から離れた面リングを生成します。負のバッファ距離は、中心へ向かう面または複数面リングを生成します。面または複数面の内部リングの場合、正のバッファ距離は、中心へ向かうバッファ・リングを生成し、負のバッファ距離は、中心から離れるバッファ・リングを生成します。

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、結果として空の図形になります。

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図形の凸包を戻します。頂点 は、図形内の 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 に似た不規則な図形が描かれています。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。凸包は、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 つの図形を入力としてとります。*ST_Difference* 関数は、1 番目の図形の、2 番目の図形が交差しない部分を戻します。この操作は、地理情報における論理 AND NOT 操作です。*ST_Difference* によって戻された図形の部分は、それ自身が図形、つまり入力となった図形と同じディメンションを持つ集合です。これらの 2 つの図形が等しい場合、つまり、それらが同じスペースを占める場合、戻される図形は空です。

各矢印の左側には、入力として *ST_Difference* に与えられた 2 つの図形があります。各矢印の右側には、*ST_Difference* の出力があります。1 番目の図形の一部に 2 番目の図形が交差する場合、1 番目の図形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図形が等しい場合、出力は空の図形です (*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。たとえば、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。

<p>ポイント / ポイント <i>nil</i></p>	<p>ポイント / ポイント 複数ポイント</p>	<p>ポイント / 複数ポイント 複数ポイント</p>
<p>複数ポイント / 複数ポイント <i>nil</i></p>	<p>複数ポイント / 複数ポイント 複数ポイント</p>	<p>折れ線 / 折れ線 複数折れ線</p>
<p>折れ線 / 折れ線 <i>nil</i></p>	<p>ポリゴン / ポリゴン <i>nil</i></p>	<p>ポリゴン / ポリゴン ポリゴン</p>

図 53. *ST_Difference*

ST_Intersection

ST_Intersection 関数は、2 つの与えられた図形の交差を定義する、図形として表される、ポイントの集合を戻します。入力として ST_Intersection に与えられた図形が交差しない場合、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図形のディメンションよりも小さい場合、ST_Intersection は空の図形を戻します。

各矢印の左側には、入力として ST_Intersection に与えられた 2 つの交差する図形があります。各矢印の右側には、ST_Intersection の出力、つまり、左側の図形によって作成された交差を表す図形があります。

下の図は、ST_Intersection の出力の例です。入力として与えられた図形がどこで交差しているかという情報を戻します。たとえば、図形 b が折れ線で図形 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マルチポイントになります。図形 a と図形 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マルチポリゴンになります。

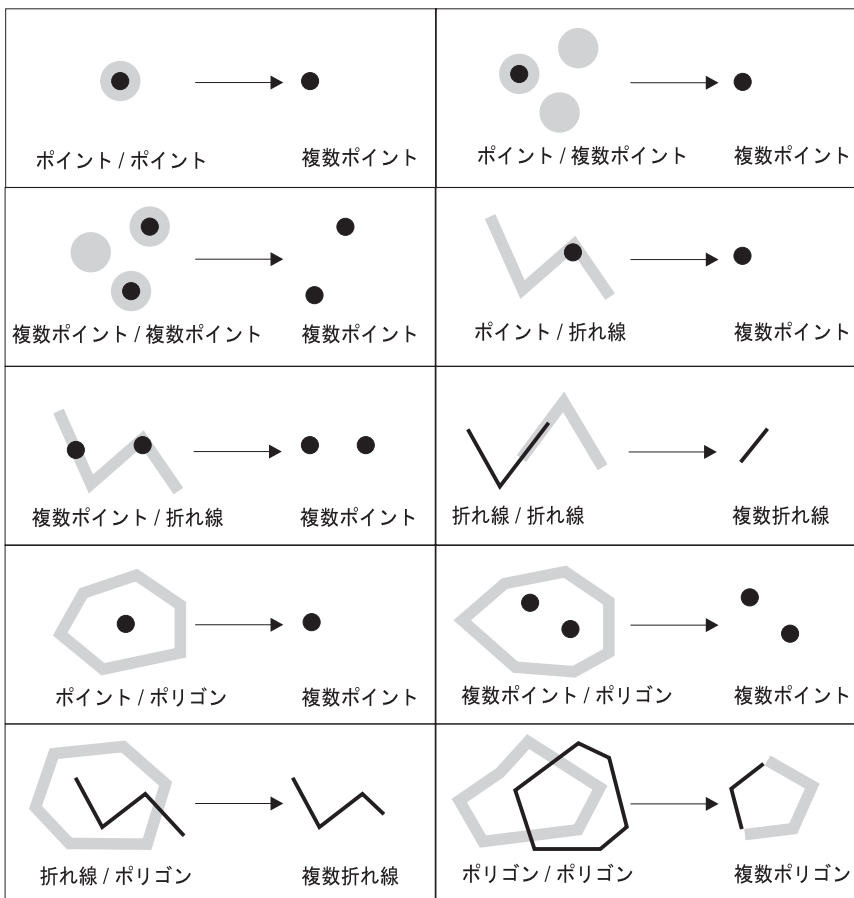


図 54. ST_Intersection

ST_SymDifference

ST_SymDifference 関数は、同じディメンションをもつ 2 つの交差する図形の対称差 (地理情報における論理 XOR 操作) を戻します。これらの図形が等しい場合、

ST_SymDifference は空の図形を戻します。図形が等しくない場合は、図形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。

複数の図形から 1 つの図形を派生させる関数

以下の関数は、複数の図形から個別の図形を派生させます。たとえば、ST_Union は、2 つの図形を単一の図形に結合します。

MBR 集約

関数 ST_BuildMBRAggr および ST_GetAggrResult の組み合わせは、列の中のすべての図形を囲む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図形の列を、単一の図形に集約します。集約を計算するとき、Z 座標と M 座標は棄てられます。

ST_Union

ST_Union 関数は、2 つの図形の共用体セットを戻します。この操作は、地理情報における論理 OR です。2 つの図形は、同じディメンションの図形でなければなりません。ST_Union は、常に結果を集約として戻します。

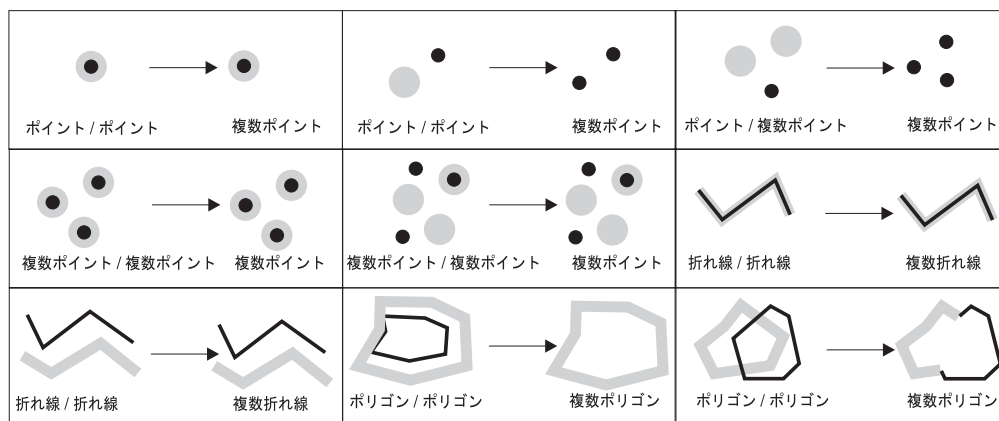


図 55. ST_Union

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、表内の図形の列を和集約として集約し、1 つの図形にします。

指標に基づいて新規図形を派生させる関数

このセクションで説明されている関数は、そのポイントが特定の指標、または特定の順序の 2 つの指標に関連付けられている図形を作成することができます。たとえば、4 から 8 までの値の指標が複数曲線のポイントと共に保管されていると仮定します。7 の値を持つ指標がどのポイントと共に保管されているかを知りたい場合、ST_FindMeasure 関数を使用して、単一の複数ポイント内のそれらのポイントに戻すことができます。

これらの関数には以下のものがあります。

- ST_FindMeasure (ST_LocateAlong と呼ばれる)
- ST_MeasureBetween (ST_LocateBetween と呼ばれる)

ST_FindMeasure (ST_LocateAlong と呼ばれる)

ST_FindMeasure (または ST_LocateAlong) は図形と指標を入力パラメーターとします。指定した指標に一致する特定の図形の複数ポイントまたは複数曲線に戻します。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、図形の境界に関して行われます。

ST_MeasureBetween (ST_LocateBetween と呼ばれる)

ST_MeasureBetween (または ST_LocateBetween) は、図形および 2 つの M 座標 (指標) を入力パラメーターとし、その図形の、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に戻します。

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。図 56 では、ポイント 3、4、5、6、7、8、9 が曲線を表しています。2 つの M 座標が 4、7 であれば、ST_MeasureBetween は曲線のポイント 4 と 7 の間の部分に戻します。

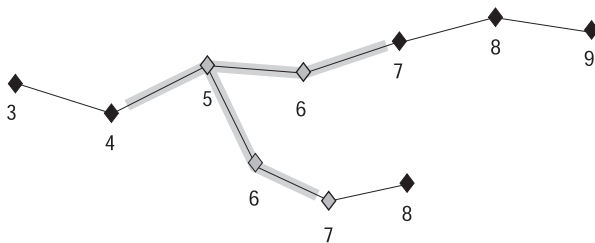


図 56. LocateBetween

既存の図形の修正フォームを作成する関数

以下の関数は、既存の図形の修正フォームを作成します。たとえば、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 は図形としきい値を入力パラメーターとし、図形の一般的特性を保持しつつ、ポイントの数を減らして、与えられた図形を表現します。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、図形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以上に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 つの図形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 番目の図形内の任意のポイントと 2 番目の図形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 番目の図形が 1 番目の図形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 つの図形のいずれかが NULL または空の場合は、NULL が戻されます。

たとえば、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。347 ページの図 57 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。

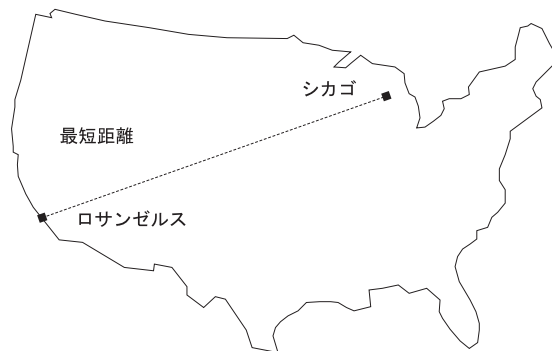


図 57. 2 つの都市の間の最短距離： `ST_Distance` は、ロサンゼルスとシカゴのロケーションの座標を入力とし、それらのロケーション間の最短距離を表す値を返すことができます。

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。

座標系間の変換

`ST_Transform` は、図形および空間参照系 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図形をトランスフォームします。異なる座標系の間で投影および変換が行われ、図形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。ここでは関数は英数字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。

- 考慮する要素: 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図形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

考慮する要素

空間処理関数を使用する場合は、以下の要素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。たとえば、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。たとえば、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

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

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・タイプである図形を空間処理関数が戻し、その図形をこのスーパー・タイプに從属するタイプの図形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。

たとえば、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 から戻された図形を 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・タイプの図形を渡すには、TREAT 演算子を使用します。前述のように、ST_Union は ST_Geometry の静的タイプの図形を戻します。これは、ST_Geometry の動的サブタイプの図形を戻すこともできます。ここでたとえば、ST_MultiPolygon の動的タイプを持つ図形を戻すとします。この場合、TREAT 演算子は、この図形を静的タイプ 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・タイプの図形を戻す場合、TREAT 演算子は通常、この図形をこのスーパー・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、図形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、またはこれに從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。たとえば、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 図形を受け付けます。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 に從属するため、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

表 56 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。

たとえば、表 56 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 56. 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

空間処理関数の考慮事項

表 56. 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 56. 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

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。

関連資料:

- 356 ページの『MBR 集約』
- 366 ページの『ST_Boundary』
- 358 ページの『ST_Area』
- 475 ページの『ST_PerpPoints』
- 477 ページの『ST_Point』
- 483 ページの『ST_PointOnSurface』
- 490 ページの『ST_Relate』
- 508 ページの『ST_Union』
- 518 ページの『和集約』

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 つの図形

EnvelopesIntersect は、最初の図形のエンベロープが 2 番目の図形のエンベロープと交差する場合に 1 を返します。それ以外の場合、0 (ゼロ) が返されます。

- 図形、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、最初の図形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 を返します。それ以外の場合、0 (ゼロ) が返されます。

構文:

```
db2gse.EnvelopesIntersect(geometry1, geometry2, rectangular-window)
```

rectangular-window:

```
x_min, y_min, x_max, y_max, srs_id
```

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、図形を表すタイプ ST_Geometry (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、図形を表すタイプ ST_Geometry (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 では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 度までの間の経度でなくてはならない。
- *x_min* は、エンベロープが第 180 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 では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 度までの間の緯度でなくてはならない。
- *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 度までの間の経度でなくてはならない。
- *x_max* は、エンベロープが第 180 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 度までの間の緯度でなくてはならない。
- *y_max* の値は *y_min* より大きくななくてはならない。

srs_id

空間参照系を一意的に識別します。空間参照系 ID は、図形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図形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、列の中のすべての図形を囲む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図形の列を、単一の図形に集約します。集約を計算するときに、Z 座標と M 座標は棄てられます。

結合するすべての図形が `NULL` の場合は、`NULL` が戻されます。図形のすべてが `NULL` または空である場合は、空の図形が戻されます。結合するすべての図形を囲む最小の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。結合するすべての図形を囲む最小の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、図形を囲む最小の長方形が `ST_Polygon` 値として戻されます。

構文:

```
▶—db2gse.ST_GetAggrResult—(—MAX—(——————)—————)—————▶
▶—db2gse.ST_BuildMBRAggr—(—geometries—)——)—————▶
```

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図形を表す、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、全選択内で地理情報列の和集約を作成できません。

- MPP 環境内。
- 全選択で `GROUP BY` 文節を使用した場合。
- 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、列内のすべての図形を囲む最大の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、すべてのポイントを囲む最大の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:

```

▶▶ db2gse.ST_AppendPoint(—curve—, —point—) ◀◀

```

パラメーター:

curve *point* が付加される曲線を表す、タイプが ST_Curve (またはそのサブタイプのいずれか) の値。

point *curve* に付加するポイントを表す、タイプが ST_Point の値。

戻りタイプ:

db2gse.ST_Curve

ST_AppendPoint

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、図形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
| えられた図形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。

| 図形がポリゴンまたは複数ポリゴンの場合は、その図形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。図形が 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:

```
db2gse.ST_Area(geometry [, unit])
```

パラメーター:

geometry

そのエリアを判別する図形を表す、タイプが ST_Geometry (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* が投影座標系、または地心から見た座標系の場合、この座標系に関連付けられた線形単位が使用されます。
- *geometry* が地理座標系で、測地地理座標系 (SRS) でない場合、この座標系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 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。

- 図形の座標系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。
- 図形の座標系が投影座標系で、かつ角度単位が指定されている。
- 図形の座標系が地理座標系で、測地 SRS でなく、かつ線形単位が指定されている。
- 図形の座標系が地理座標系かつ測地 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:

地理情報分析者は、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 関数を図形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)
```

ST_Area

```
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	+1.000000000000000E+002
2	+2.000000000000000E+002
3	+2.500000000000000E+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	+1.000000000000000E+002	+9.29034116132748E+000	+3.58702077598427E-006
2	+2.000000000000000E+002	+1.85806823226550E+001	+7.17404155196855E-006
3	+2.500000000000000E+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_GEODETTIC_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_GEODETTIC_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 およびエリアを平方メートル、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


```
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'01010000000000000000000024400000000000003440'
```

関連資料:

- 532 ページの『事前割り当てバイナリー (WKB) 表記』

ST_AsGML

ST_AsGML は図形を入力パラメーターとし、ジオグラフィー・マークアップ言語を使用してその図形を表現したものを戻します。

与えられた図形が 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:

```
▶▶db2gse.ST_AsGML(—geometry—)▶▶
```

パラメーター:

geometry

対応する GML 表記に変換される、タイプが ST_Geometry (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、図形列から 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))
```

ST_AsGML

```
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、ID および図形の GML 表記をリストします。図形は ST_AsGML 関数により GML フラグメントに変換されます。

```
SELECT id, cast(ST_AsGML(geometry) AS varchar(110)) AS gml_fragment
FROM sample_points
WHERE id = 1100
```

結果:

The SELECT statement returns the following result set:

ID	GML_FRAGMENT
1100	<gml:Point srsName";EPSG:4269";><gml:coord> <gml:X>10</gml:X><gml:Y>20</gml:Y> </gml:coord></gml:Point>

関連資料:

- 309 ページの『図形値をデータ交換フォーマットに変換する空間処理関数』
- 534 ページの『ジオグラフィー・マークアップ言語 (GML) 表記』

ST_AsShape

ST_AsShape は形状を入力パラメーターとして取り、その ESRI 形状表記を戻します。

与えられた図形が 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:

▶▶—db2gse.ST_AsShape—(*geometry*)—▶▶

パラメーター:

geometry

対応する ESRI 形状表記に変換される、タイプが ST_Geometry (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 表の図形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、図形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

一で図形を表示する場合 (この場合、図形は 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 ファイル用に図形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'01000000000000000000000024400000000000003440'
```

関連資料:

- 533 ページの『形状表記』

ST_AsText

ST_AsText は、図形を入力パラメーターとして取り、その良く知られたテキスト表記を戻します。

与えられた図形が 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:

```
▶▶ db2gse.ST_AsText(—geometry—)▶▶
```

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、タイプが ST_Geometry (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。

ST_AsText

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、図形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、図形の空間データ・タイプおよび WKT 表記をリストします。図形は 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	LINSTRING (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))

関連資料:

- 527 ページの『事前割り当てテキスト (WKT) 表記』

ST_Boundary

ST_Boundary は図形を入力パラメーターとし、その境界を新しい図形として戻します。結果の図形は、与えられた図形の空間参照系で表現されます。

与えられた図形が、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、または与えられた図形が空の場合、結果はタイプ ST_Point の空の図形になります。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、与えられた図形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。与えられた図形が NULL の場合は NULL が戻されます。

可能な場合には、戻される図形のタイプは ST_Point、ST_LineString、または ST_Polygon になります。たとえば、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:

```
db2gse.ST_Boundary(geometry)
```

パラメーター:

geometry

タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。この図形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、複数の図形を作成し、それぞれの図形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, 80.00000000 130.00000000, 80.00000000 140.00000000, 70.00000000 140.00000000, 70.00000000 130.00000000))
3	MULTIPOINT (60.00000000 60.00000000, 65.00000000 60.00000000, 65.00000000 70.00000000, 70.00000000 70.00000000)
4	MULTIPOINT (50.00000000 50.00000000, 55.00000000 50.00000000, 55.00000000 60.00000000, 60.00000000 60.00000000, 65.00000000 60.00000000, 65.00000000 70.00000000, 70.00000000 70.00000000, 80.00000000 80.00000000, 85.00000000 80.00000000, 85.00000000 90.00000000, 90.00000000 90.00000000)
5	POINT EMPTY

ST_Buffer

ST_Buffer は、図形、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図形を囲む図形を戻します。結果の図形の境界上の各ポイントは、指定された距離だけ、与えられた図形から離れています。結果の図形は、与えられた図形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、入力された図形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。

結果の図形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。たとえば、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。

与えられた図形が 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:

```
▶▶ db2gse.ST_Buffer(geometry, distance, unit)
```

パラメーター:

geometry

周りにバッファーを作成する図形を表す、タイプが ST_Geometry (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まります。

- *geometry* が投影座標系、または地心から見た座標系の場合、この座標系に関連付けられた線形単位がデフォルトになります。
- *geometry* が地理座標系で、測地地理座標系 (SRS) でない場合、この座標系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 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。

- 図形の座標系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。
- 図形の座標系が投影座標系で、かつ角度単位が指定されている。
- 図形の座標系が地理座標系で、測地 SRS でなく、かつ線形単位が指定されている。
- 図形の座標系が地理座標系かつ測地 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。

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 1:

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

ST_Buffer

```
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 2:

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 3:

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))
```

関連資料:

- 218 ページの『DB2 Geodetic Extender によってサポートされる空間処理関数』
- 307 ページの『DB2GSE.ST_UNITS_OF_MEASURE カタログ・ビュー』

ST_Centroid

ST_Centroid は図形を入力パラメーターとし、図形の中心を戻します。図形の中心とは、与えられた図形の最小の長方形範囲の中心のポイントです。結果のポイント は、与えられた図形の空間参照系で表現されます。

与えられた図形が 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:

```
►—db2gse.ST_Centroid—(—geometry—)—————►
```

パラメーター:*geometry*

その中心を判別する図形を表す、タイプが ST_Geometry (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 つの図形を作成し、その図心を見つけてみます。

```
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。結果の図形は、与えられた図形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:

```
►►—db2gse.ST_ChangePoint—(—curve—,—old_point—,—new_point—)—————►►
```

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、タイプが ST_Curve (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、タイプが ST_Point の値。

new_point

曲線内の、*old_point* で示されたポイントの新しいロケーションを表す、タイプが ST_Point 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:

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 1:

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
```

結果:

```
NEW
-----
LINESTRING ( 10.00000000 10.00000000, 6.00000000 6.00000000, 0.00000000
0.00000000, 10.00000000 0.00000000, 6.00000000 6.00000000, 0.00000000
10.00000000)
```

例 2:

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 つの図形を入力パラメーターとし、2 番目の図形が 1 番目の図形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) を戻して最初の図形に 2 番目の図形が完全には含まれていないことを示します。

与えられた図形のいずれかが NULL 値または空の場合は、NULL 値が戻されま

非測地データの場合、2 番目の図形が 1 番目の図形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、両方の図形が同じ測地参照系 (SRS) で表現される必要があります。

構文:

```
►►—db2gse.ST_Contains—(—geometry1—,—geometry2—)—————►►
```

パラメーター:

geometry1

geometry2 を完全に含むかどうかをテストする図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

ST_Contains

geometry2

geometry1 内に完全に含まれるかどうかをテストする図形を表す、タイプが ST_Geometry (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、両方の図形が測地で、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 1:

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 2:

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

関連資料:

- 510 ページの『ST_Within』

ST_ConvexHull

ST_ConvexHull は図形を入力パラメーターとし、その凸包を戻します。

結果の図形は、与えられた図形の空間参照系で表現されます。

可能な場合には、戻される図形のタイプは ST_Point、ST_LineString、または ST_Polygon になります。たとえば、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。

与えられた図形が 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:

```

▶▶—db2gse.ST_ConvexHull—(—geometry—)—————▶▶

```

パラメーター:

geometry

凸包を計算する図形を表す、タイプが ST_Geometry の値、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

```

ST_ConvexHull

```
('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図形の凸包を計算し、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 は図形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。

与えられた図形が 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。図形が 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:

```
►►db2gse.ST_CoordDim(—geometry—)◄◄
```

パラメーター:

geometry

ディメンション数を検索しようとする図形を表す、タイプが ST_Geometry (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、複数の図形を作成し、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 つの図形を入力パラメーターとし、1 番目の図形が 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 番目の図形が 1 番目の図形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。

ST_Crosses

1 番目の図形がポリゴンまたは複数ポリゴンの場合、または 2 番目の図形がポイントまたは複数ポイントの場合、あるいは図形のいずれかが NULL または空である場合は、NULL が戻されます。2 つの図形の交差が、指定された 2 つの図形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図形になり、結果の図形が指定された 2 つの図形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:

```
▶▶—db2gse.ST_Crosses—(—geometry1—,—geometry2—)—————▶▶
```

パラメーター:

geometry1

geometry2 との交わりをテストする図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

geometry2

geometry1 が交わっているかどうかをテストする図形を表す、タイプが ST_Geometry (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、作成された図形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

関連資料:

- 318 ページの『地勢を比較する関数』

ST_Difference

ST_Difference は、2 つの図形を入力パラメーターとし、1 番目の図形が 2 番目と交わらない部分を戻します。

2 つの図形は、同じディメンションの図形でなければなりません。いずれかの図形が NULL の場合は NULL が戻されます。1 番目の図形が空の場合、タイプ ST_Point の空の図形が戻されます。2 番目の図形が空の場合は、1 番目の図形が変更されずに戻されます。

非測地データの場合、2 番目の図形が 1 番目の図形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、両方の図形が同じ測地参照系 (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:

```
►—db2gse.ST_Difference—(—geometry1—,—geometry2—)—————►
```

パラメーター:

geometry1

差を *geometry2* に計算するために使用する、1 番目の図形を表す、タイプが ST_Geometry の値。

geometry2

geometry1 との差を計算するために使用する、2 番目の図形を表す、タイプが ST_Geometry の値。

測地データの制約事項

- 両方の図形が測地で、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

戻される図形のディメンションは、入力された図形と同じになります。

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
```

ST_Difference

```
(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 は図形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。

与えられた図形が空の場合は -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。与えられた図形が 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:

▶▶ db2gse.ST_Dimension(—geometry—) ▶▶

パラメーター:

geometry

そのディメンションを戻す図形を表す、タイプが ST_Geometry の値。

戻りタイプ:

INTEGER

例:

この例は、複数の異なる図形を作成し、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 つの図形を入力パラメーターとし、与えられた図形が交差しない場合は 1 を戻します。図形が交差する場合は、0 (ゼロ) が戻されます。

ST_Disjoint

2 番目の図形が 1 番目の図形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。

与えられた図形のいずれかが 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:

```
db2gse.ST_Disjoint(geometry1, geometry2)
```

パラメーター:

geometry1

geometry2 との交差をテストする図形を表す、タイプが ST_Geometry の値。

geometry2

geometry1 との交差をテストする図形を表す、タイプが ST_Geometry の値。

戻りタイプ:

INTEGER

例:

次のコードは SAMPLE_GEOMETRIES 表に複数の図形を作成します。

```
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 1:

この例は、最初のポリゴンがどの図形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 2:

この例は、3 番目のポリゴンがどの図形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 3:

この例は、2 番目の折れ線がどの図形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

関連資料:

- 318 ページの『地勢を比較する関数』

ST_Distance

ST_Distance は、2 つの図形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 番目の図形内の任意のポイントと 2 番目の図形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 つの図形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。詳細については、171 ページの『測地距離』を参照してください。

2 つの図形のいずれかが 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 番目の図形が 1 番目の図形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、両方の図形が同じ測地参照系 (SRS) で表現される必要があります。

ST_Distance

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_Distance(geometry1, geometry2, [unit])
```

パラメーター:

geometry1

geometry2 との距離を計算するために使用する図形を表す、タイプが ST_Geometry の値。

geometry2

geometry1 との距離を計算するために使用する、図形を表す、タイプが ST_Geometry 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、両方の図形が同じ測地 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* が投影座標系、または地心から見た座標系の場合、この座標系に関連付けられた線形単位がデフォルトになります。
- *geometry1* が地理座標系で、測地地理座標系 (SRS) でない場合、この座標系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。

- 図形の座標系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。
- 図形の座標系が投影座標系で、かつ角度単位が指定されている。
- 図形の座標系が地理座標系で、測地 SRS でなく、かつ線形単位が指定されている。
- 図形の座標系が地理座標系かつ測地 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:

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 1:

次の SELECT ステートメントは、SAMPLE_GEOMTRIES1 表と SAMPLE_GEOMTRIES2 表にある各種の図形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 sg2_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	SG2_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 2:

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図形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 sg2_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:

ST_Distance

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 3:

次の SELECT ステートメントは、各種の図形間の距離をキロメートルで計算します。

```
SAMPLE_GEOMETRIES1 and SAMPLE_GEOMETRIES2 tables.  
SELECT sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,  
       sg2.id AS sg2_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

関連資料:

- 318 ページの『地勢を比較する関数』

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 ファイルはジオコーダー・データ CD 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:

```

▶db2gse.ST_Edge_GC_USA(—street—,—city—,—state—,—zip—,—srs_id—,—
▶spelling_sens—,—min_match_score—,—side_offset—,—side_offset_units—,—end_offset—,—
▶base_map—,—locator_file—)

```

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、地理座標系 GCS_NORTH_AMERICAN_1983 に基づいて投影座標系を使用する、既存の空間参照系を示すか、またはこの地理座標系自体 (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・ファイルから取られます。ロケータ・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・ファイルから取られます。ロケータ・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
- Decimal degrees
- 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・ファイルから取られます。ロケータ・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けば使用できます。

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
```

ST_Edge_GC_USA

結果:

```
1
-----
POINT ( -77.02829300 38.90049000)
POINT ( -121.94507200 37.28766700)
```

例 2:

この例では、投影座標系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)

NetBIOS
```

注: 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に戻します。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:

▶▶—db2gse.ST_EndPoint—(—curve—)————▶▶

パラメーター:

curve 最後のポイントを戻す図形を表す、タイプが ST_Curve の値。

戻りタイプ:

db2gse.ST_Point

例:

SELECT ステートメントは、SAMPLE_LINES 表内のそれぞれの図形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)

関連資料:

- 482 ページの『ST_PointN』

ST_Envelope

ST_Envelope は図形を入力パラメーターとし、図形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。

与えられた図形がポイント、水平線、または垂直線の場合、与えられた図形よりも少し大きい長方形が戻されます。それ以外の場合、図形を囲む最小の長方形がエンベロープとして戻されます。与えられた図形が NULL または空の場合は、NULL が戻されます。すべての図形について、それを囲む正確な最小の長方形を戻すには、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、図形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。

ST_Envelope

構文:

```
▶▶ db2gse.ST_Envelope(geometry)▶▶
```

パラメーター:

geometry

エンベロープを戻す図形を表す、タイプが ST_Geometry 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、複数の図形を作成し、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、エンベロープは図形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))
```

関連資料:

- 442 ページの『ST_MBR』

ST_EnvIntersects

ST_EnvIntersects は 2 つの図形を入力パラメーターとし、2 つの図形のエンベロープが交差する場合は 1 を返します。それ以外の場合、0 (ゼロ) が返されます。

2 番目の図形が 1 番目の図形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。

いずれかの図形が NULL または空の場合は、NULL 値が返されます。

構文:

```
▶▶—db2gse.ST_EnvIntersects—(—geometry1—,—geometry2—)————▶▶
```

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、図形を表すタイプ ST_Geometry (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、図形を表すタイプ ST_Geometry (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:

ST_EnvIntersects

ID	ID	INTERSECTS	ENVELOPE_INTERSECTS
-----	-----	-----	-----
	1	2	0
			1

ST_EqualCoordsys

ST_EqualCoordsys は 2 つの座標系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 を返します。それ以外の場合、整数値 0 (ゼロ) が返されます。座標系定義は、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。

与えられた座標系定義のいずれかが NULL の場合は、NULL が返されます。

構文:

```
▶▶db2gse.ST_EqualCoordsys(—coordinate_system1—,—coordinate_system2—)▶▶
```

パラメーター:

coordinate_system1

coordinate_system2 と比較される 1 番目の座標系を定義する VARCHAR(2048) の値。

coordinate_system2

coordinate_system1 と比較される 2 番目の座標系を定義する VARCHAR(2048) 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系を比較し、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
```

関連資料:

• 297 ページの『DB2GSE.ST_COORDINATE_SYSTEMS カタログ・ビュー』

ST_Equals

ST_Equals は 2 つの図形を入力パラメーターとし、図形が等しい場合は 1 を返します。それ以外の場合、0 (ゼロ) が返されます。図形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 番目の図形が 1 番目の図形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 つの図形のいずれかが NULL の場合は、NULL が返されます。

構文:

```
►►db2gse.ST_Equals(—geometry1—,—geometry2—)—————▶▶
```

パラメーター:

geometry1

geometry2 と比較される図形を表す、タイプが ST_Geometry の値。

geometry2

geometry1 と比較される図形を表す、タイプが ST_Geometry の値。

戻りタイプ:

INTEGER

例:

例 1:

この例は、異なる順序で座標系を持つ 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:

ST_Equals

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 つの図形を作成します。ST_Equals 関数を使用して図形を比較すると、0 (ゼロ) が戻され、これらの図形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 つの図形を作成していますが、両方とも同じ図形を表します。ST_Equals はこれらの図形を比較し、両方の図形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

関連資料:

- 318 ページの『地勢を比較する関数』

ST_EqualsSRS

ST_EqualsSRS 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、スケール因数、および座標系が比較されます。

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:

▶▶ db2gse.ST_EqualSRS(—srs_id1—,—srs_id2—)▶▶

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値 (タイプ INTEGER)。

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値 (タイプ INTEGER)。

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、同じ座標系を参照します。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 を返し、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
```

関連資料:

- 305 ページの『DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビュー』

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:

ST_ExteriorRing

▶▶—db2gse.ST_ExteriorRing—(—polygon—)————▶▶

パラメーター:

polygon

外部リングを戻すポリゴンを表す、タイプが ST_Polygon 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)

関連資料:

- 366 ページの『ST_Boundary』

ST_FindMeasure または ST_LocateAlong

ST_FindMeasure または ST_LocateAlong は図形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、指定された図形の指定された指標そのものをもつ図形部分の、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、図形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、空の図形が戻されます。他のすべての図形については、与えられた指標が図形内の最小の指標より低い場合、または図形内の最高の指標より高い場合は、空の図形が戻されます。与えられた図形が 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:

```
db2gse.ST_FindMeasure(geometry, measure)
```

パラメーター:

geometry

M 座標 (指標) に *measure* が含まれる部分を探したい図形を表す、タイプが ST_Geometry (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、タイプ DOUBLE の値。

戻りタイプ:

db2gse.ST_Geometry

例:

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 (サンプルの図形を保管する列) 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 1:

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図形) 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
```

ST_FindMeasure または ST_LocateAlong

結果:

```
ID      MEASURE_7
-----
1 POINT M ( 3.50000000 3.50000000 7.00000000)
2 POINT EMPTY
```

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、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)
```

関連資料:

- 444 ページの『ST_MeasureBetween、ST_LocateBetween』

ST_Generalize

ST_Generalize は図形としきい値を入力パラメーターとし、図形の一般的特性を保持しつつ、ポイントの数を減らして、与えられた図形を表現します。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、図形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以上に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。結果の図形は、与えられた図形の空間参照系にあります。

指定された図形が空の場合、タイプ ST_Point の空の図形が戻されます。指定された図形またはしきい値が 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:

```
▶▶—db2gse.ST_Generalize—(—geometry—,—threshold—)————▶▶
```

パラメーター:

geometry

線の単純化を適用する図形を表す、タイプが ST_Geometry (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。この値は 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、一般化された図形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。

(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 1:

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 2:

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

ST_GeomCollection は、次の入力の 1 つから図形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
- ジオグラフィー・マークアップ言語 (GML) 表記

結果の図形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:

```

▶▶ db2gse.ST_GeomCollection ( ( wkt | wkb | shape | gml ) [, srs_id ] )

```

パラメーター:

- wkt* 結果の図形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* 結果の図形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* 結果の図形集合の ESRI 形状表記を表す、タイプが BLOB(2G) の値。
- gml* ジオグラフィー・マークアップ言語 (GML) を使用した、結果の図形集合を表す、タイプが CLOB(2G) の値。
- srs_id* 結果の図形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 データ・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、またジオグラフィー・マークアップ言語 (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)

関連資料:

- 527 ページの『事前割り当てテキスト (WKT) 表記』
- 532 ページの『事前割り当てバイナリー (WKB) 表記』
- 533 ページの『形状表記』
- 534 ページの『ジオグラフィー・マークアップ言語 (GML) 表記』

ST_GeomCollFromTxt

ST_GeomCollFromTxt は、図形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、対応する図形集合を戻します。

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:

```
db2gse.ST_GeomCollFromTxt( (wkt [, srs_id] ) )
```

パラメーター:

wkt 結果の図形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 結果の図形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)))
```

関連資料:

- 402 ページの『ST_GeomCollection』

ST_GeomCollFromWKB

ST_GeomCollFromWKB は、入力パラメーターとして、図形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、対応する図形集合を戻します。

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 が戻されません。

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:

```
db2gse.ST_GeomCollFromWKB(—wkb— [,—srs_id—])
```

パラメーター:

wkb 結果の図形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 結果の図形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

ST_GeomCollFromWKB

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図形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 の図形集合を持つ行を 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))
```

関連資料:

- 532 ページの『事前割り当てバイナリー (WKB) 表記』

ST_Geometry

ST_Geometry は、次の入力の 1 つから図形を作成します。

- 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
- ジオグラフィー・マークアップ言語 (GML) 表記

結果の図形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。

結果の図形の動的タイプは、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:

```
db2gse.ST_Geometry( ( wkt
                    | wkb
                    | shape
                    | gml
                    ) , srs_id )
```

パラメーター:

- wkt* 結果の図形の事前割り当てテキスト表記が入る、タイプ CLOB(2G) の値。
- wkb* 結果の図形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* 結果の図形の ESRI 形状表記を表す、タイプが BLOB(2G) の値。
- gml* ジオグラフィー・マークアップ言語 (GML) を使用した、結果の図形を表す、タイプが CLOB(2G) の値。
- srs_id* 結果の図形の空間参照系を識別する、タイプが 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_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、またはジオグラフィー・マークアップ言語 (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、各種の図形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>
```

ST_Geometry

```
</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)

関連資料:

- 527 ページの『事前割り当てテキスト (WKT) 表記』

ST_GeometryN

ST_GeometryN は、図形の集合と 1 つの索引を入力パラメーターとし、集合の中から、索引で指定された図形を戻します。結果の図形は、与えられた図形集合の空間参照系で表現されます。

与えられた図形集合が 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図形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:

```
▶▶ db2gse.ST_GeometryN(—collection—, —index—) ▶▶
```

パラメーター:

collection

その中にある *n* 番目の図形を探す図形集合を表す、タイプが ST_GeomCollection (またはそのサブタイプの 1 つ) の値。

index *collection* から戻される *n* 番目の図形を示す、タイプ INTEGER の値。

index が 1 より小さいか集合内の図形の数より大きい場合は、NULL が戻され、警告 (SQLSTATE 01HS0) が出されます。

戻りタイプ:

db2gse.ST_Geometry

例:

次のコードは、図形集合の中の 2 番目の図形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))

```

関連資料:

- 467 ページの『ST_NumGeometries』

ST_GeometryType

ST_GeometryType は図形を入力パラメーターとし、その図形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:

```

▶▶—db2gse.ST_GeometryType—(—geometry—)————▶▶

```

パラメーター:

geometry

図形タイプを戻す、タイプ ST_Geometry の値。

戻りタイプ:

VARCHAR(128)

例:

次のコードは、図形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)

```

ST_GeometryType

```
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_MULTIPPOINT"

ST_GeomFromText

ST_GeomFromText は、図形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、対応する図形を戻します。

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:

```
db2gse.ST_GeomFromText(wkt, srs_id)
```

パラメーター:

wkt 結果の図形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 結果の図形の空間参照系を識別する、タイプが 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_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 の図形を持つ行を 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))

関連資料:

- 527 ページの『事前割り当てテキスト (WKT) 表記』

ST_GeomFromWKB

ST_GeomFromWKB は、図形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、対応する図形を戻します。

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 が戻されません。

この機能としては、ST_Geometry の使用をお勧めします。

構文:

```

▶▶ db2gse.ST_GeomFromWKB ( ( wkb [ , srs_id ] ) )

```

パラメーター:

wkb 結果の図形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 結果の図形の空間参照系を識別する、タイプが 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。

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)。

ST_GeomFromWKB

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 表記の図形を持つレコードを 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))

関連資料:

- 532 ページの『事前割り当てバイナリー (WKB) 表記』

ST_GetIndexParms

ST_GetIndexParms は、地理情報索引の ID または地理情報列の ID のいずれかを入力パラメーターとし、索引または地理情報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:

```

▶▶ db2gse.ST_GetIndexParms(—————)
└──┬──index_schema──,──index_name————┬──)
    │└──table_schema──,──table_name──,──column_name──┬──,──grid_size_number──┘
  
```

パラメーター:*index_schema*

修飾されていない名前 *index_name* を持つ地理情報索引が属するスキーマを示す、タイプが VARCHAR(128) 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、地理情報索引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地理情報索引の、修飾されていない名前が入る、タイプが VARCHAR(128) 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、タイプが VARCHAR(128) 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、地理情報索引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

地理情報列 *column_name* を持つ表の、修飾されていない名前を含む、タイプが VARCHAR(128) 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

列の地理情報索引の索引パラメーターが戻される、表 *table_schema.table_name* の列を示す、タイプが VARCHAR(128) 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。

列に地理情報索引が定義されていない場合は、エラーが起きます (SQLSTATE 38SQ0)。

grid_size_number

値を戻させたいパラメーターを識別する、タイプが DOUBLE 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、エラーが戻されず (SQLSTATE 38SQ1)。

戻りタイプ:

DOUBLE (*grid_size_number* が指定されている場合)

ST_GetIndexParms

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:

次のコードは、地理情報列と地理情報索引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、地理情報索引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 1:

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、地理情報グリッド索引の 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
-----
+1.000000000000000E+000
```

```
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 2)
```

結果:

```
1
-----
+1.000000000000000E+001
```

```
VALUES ST_GetIndexParms('SCH', 'IDX', 3)
```

結果:

```
1
-----
+1.000000000000000E+003
```

例 2:

この例は、地理情報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      +1.000000000000000E+000
2      +1.000000000000000E+001
3      +1.000000000000000E+003
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'IDX') ) AS t

```

結果:

```

ORDINAL      VALUE
-----
1      +1.000000000000000E+000
2      +1.000000000000000E+001
3      +1.000000000000000E+003

```

関連概念:

- 104 ページの『空間グリッド・インデックス』

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、内部図形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、警告条件が起きます (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:

```

▶▶—db2gse.ST_InteriorRingN(—polygon—, —index—)—————▶▶

```

パラメーター:

polygon

index で指定された内部リングを戻す図形を表す、値 (タイプ ST_Polygon)。

index *n* 番目の内部リングを示すことを示す値 (タイプ INTEGER)。 *index* で識別される内部リングがない場合は、警告条件が起きます (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))

```

ST_InteriorRingN

```
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)
```

関連資料:

- 397 ページの『ST_ExteriorRing』
- 468 ページの『ST_NumInteriorRing』

ST_Intersection

ST_Intersection は 2 つの図形を入力パラメーターとし、与えられた 2 つの図形の交差を表す図形を戻します。交差は、1 番目の図形の 2 番目の図形と重なり合う部分です。結果の図形は、1 番目の図形の空間参照系で表現されます。

可能な場合には、戻される図形のタイプは ST_Point、ST_LineString、または ST_Polygon になります。たとえば、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 つの図形のいずれかが NULL の場合は、NULL が戻されます。

非測地データの場合、2 番目の図形が 1 番目の図形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、両方の図形が同じ測地参照系 (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:

```
db2gse.ST_Intersection(geometry1,geometry2)
```

パラメーター:

geometry1

geometry2 との交差を計算する、1 番目の図形を表す、タイプが ST_Geometry (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 番目の図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

測地データの場合は、両方の図形が同じ測地 SRS で表現される必要があります。

戻りタイプ:

db2gse.ST_Geometry

戻される図形のディメンションは、測地データの折れ線を除き、入力された図形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、あるいはマルチ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、複数の異なる図形を作成し、次に最初の図形との交差 (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	LINSTRING (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	LINSTRING (30.00000000 30.00000000, 50.00000000 50.00000000)

5 record(s) selected.

ST_Intersects

ST_Intersects は 2 つの図形を入力パラメーターとし、与えられた図形が交差する場合は 1 を返します。図形が交差しない場合は、0 (ゼロ) が返されます。

2 つの図形のいずれかが NULL または空の場合は、NULL が返されます。

非測地データの場合、2 番目の図形が 1 番目の図形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、両方の図形が同じ測地参照系 (SRS) で表現される必要があります。

構文:

```
►►—db2gse.ST_Intersects—(—geometry1—,—geometry2—)—————►►
```

パラメーター:

geometry1

geometry2 との交差をテストする図形を表す、タイプが ST_Geometry (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図形を表す、タイプが ST_Geometry (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、両方の図形が測地で、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 表にある各種の図形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

関連資料:

- 318 ページの『地勢を比較する関数』

ST_Is3d

ST_Is3d は図形を入力パラメーターとし、与えられた図形が Z 座標を持つ場合、1 を返します。それ以外の場合、0 (ゼロ) が返されます。

与えられた図形が NULL または空の場合は、NULL が返されます。

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:

```
▶▶—db2gse.ST_Is3D—(—geometry—)————▶▶
```

パラメーター:

geometry

Z 座標の存在をテストする図形を表す、タイプが ST_Geometry (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) を持つ図形、または持たない図形をいくつか作成しています。次に ST_Is3d を使用して、どの図形に 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:

```
▶▶ db2gse.ST_IsClosed(—curve—)—————▶▶
```

パラメーター:

curve テストする曲線または複数曲線を表す、タイプが ST_Curve または ST_MultiCurve (またはこれらのサブタイプ) 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。

ST_IsClosed

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 は図形を入力パラメーターとし、与えられた図形が空の場合、1 を返します。それ以外の場合、0 (ゼロ) が返されます。図形を定義するポイントを 1 つも持たない図形は、空です。

与えられた図形が NULL の場合は NULL が返されます。

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:

```
▶▶ db2gse.ST_IsEmpty(—geometry—)▶▶
```

パラメーター:

geometry

テストする図形を表す、タイプが ST_Geometry (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 つの図形を作成し、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 は図形を入力パラメーターとし、与えられた図形が M 座標 (指標) を持つ場合、1 を返します。それ以外の場合、0 (ゼロ) が返されます。

与えられた図形が NULL または空の場合は、NULL が返されます。

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:

```
db2gse.ST_IsMeasured(geometry)
```

パラメーター:

geometry

M 座標 (指標) の存在をテストする図形を表す、タイプが ST_Geometry (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) を持つ図形、または持たない図形をいくつか作成しています。次に ST_IsMeasured を使用して、どの図形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))
```

ST_IsMeasured

```
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:

▶▶ db2gse.ST_IsRing(*curve*) ▶▶

パラメーター:

curve テストする曲線を表す、タイプが ST_Curve (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

関連資料:

- 420 ページの『ST_IsClosed』
- 425 ページの『ST_IsSimple』

ST_IsSimple

ST_IsSimple は図形を入力パラメーターとし、与えられた図形が単純な場合、1 を返します。それ以外の場合、0 (ゼロ) が返されます。

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。

与えられた図形が空の場合は 1 が返されます。これが NULL の場合は、NULL が返されます。

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:

```

▶▶ db2gse.ST_IsSimple(—geometry—) ◀◀

```

パラメーター:

geometry

テストする図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

ST_IsSimple

INTEGER

例:

この例では、いくつかの図形を作成し、それらが単純であるかどうかをチェックします。ID 4 の図形は、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 の図形は、折れ線がそれ自体を横切っているため、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 は図形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。図形は、構造化されたタイプのすべての属性が図形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。

与えられた図形が 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:

▶▶—db2gse.ST_IsValid—(—geometry—)————▶▶

パラメーター:

geometry

タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

INTEGER

例:

この例は複数の図形を作成し、ST_IsValid を使用してそれらの図形が有効であるかチェックします。ST_Geometry のような作成ルーチンは、無効な図形の作成を許さないなので、図形はすべて有効です。

```
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。

ST_Length

構文:

```
db2gse.ST_Length(curve [, unit])
```

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* が投影座標系、または地心から見た座標系の場合、この座標系に関連付けられた線形単位がデフォルトになります。
- *curve* が地理座標系で、測地地理座標系 (SRS) でない場合、この座標系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 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。

- *curve* の座標系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* の座標系が投影座標系で、かつ角度単位が指定されている。
- *curve* が地理座標系で、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:

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 1:

次の SELECT ステートメントは、SAMPLE_GEOMETRIES 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 2:

次の SELECT ステートメントは、SAMPLE_GEOMETRIES 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、対応する折れ線に戻します。

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:

```
db2gse.ST_LineFromText(wkt, srs_id)
```

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、タイプが CLOB(2G) の値。

srs_id 結果の折れ線の空間参照系を識別する、タイプが 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、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。

ST_LineFromText

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 ( 850.000000000 250.000000000, 850.000000000 850.000000000)
1111 LINESTRING EMPTY
```

関連資料:

- 527 ページの『事前割り当てテキスト (WKT) 表記』

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:

```
▶▶ db2gse.ST_LineFromWKB ( ( wkb ) [ , srs_id ] ) ▶▶
```

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、タイプが BLOB(2G) の値。

srs_id 結果の折れ線の空間参照系を識別する、タイプが 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、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)
```

関連資料:

- 532 ページの『事前割り当てバイナリー (WKB) 表記』

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
- ジオグラフィー・マークアップ言語 (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:

```
db2gse.ST_LineString( ( wkt | wkb | shape | gml ) [ , -srs_id ] )
```

パラメーター:

ST_LineString

- wkt* 結果のポリゴンの事前割り当てテキスト表記が入る、タイプが CLOB(2G) 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、タイプが BLOB(2G) の値。
- shape* 結果のポリゴンの ESRI 形状表記を表す、タイプが BLOB(2G) の値。
- gml* ジオグラフィー・マークアップ言語 (GML) を使用した、結果のポリゴンを表す、タイプが CLOB(2G) の値。
- srs_id* 結果のポリゴンの空間参照系を識別する、タイプが 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、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)
```

関連資料:

- 309 ページの『図形値をデータ交換フォーマットに変換する空間処理関数』
- 527 ページの『事前割り当てテキスト (WKT) 表記』

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:

```
►► db2gse.ST_LineStringN(—multi_linestring—, —index—)—————►►
```

パラメーター:

multi_linestring

index で識別された折れ線に戻す複数折れ線を表す、タイプが ST_MultiLineString の値。

index *multi_linestring* から戻される *n* 番目の折れ線を指定する、タイプが INTEGER 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 が戻され、警告条件 (SQLSTATE 01HS0) が出されます。

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 番目の図形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:

ST_LineStringN

```
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)
```

関連資料:

- 469 ページの『ST_NumLineStrings』

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:

```
db2gse.ST_M(point, m_coordinate)
```

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 の新しい M 座標を表す値 (タイプは DOUBLE)。

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:

以下の例は、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 1:

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	+5.000000000000000E+000
2	+4.000000000000000E+000
3	+7.000000000000000E+000

例 2:

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)

関連資料:

- 514 ページの『ST_X』
- 515 ページの『ST_Y』
- 517 ページの『ST_Z』

ST_MaxM

ST_MaxM は図形を入力パラメーターとし、その最大 M 座標を戻します。

与えられた図形が 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:

```
db2gse.ST_MaxM(geometry)
```

パラメーター:

geometry

最大 M 座標を戻す、タイプ ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

DOUBLE

例:

次の例は、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 1:

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 2:

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

関連概念:

- 437 ページの『ST_MaxX』

関連資料:

- 439 ページの『ST_MaxY』
- 440 ページの『ST_MaxZ』
- 447 ページの『ST_MinM』

ST_MaxX

ST_MaxX は図形を入力パラメーターとし、その最大 X 座標を戻します。

与えられた図形が 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:

```
▶▶ db2gse.ST_MaxX(—geometry—)▶▶
```

パラメーター:

geometry

最大 X 座標を戻す、タイプ ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

DOUBLE

例:

次の例は、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、特定の地理情報列に保管された図形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 1:

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。

ST_MaxX

```
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 2:

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 3:

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地理情報エクステントと、図形の実際の地理情報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

関連資料:

- 435 ページの『ST_MaxM』
- 439 ページの『ST_MaxY』
- 440 ページの『ST_MaxZ』
- 448 ページの『ST_MinX』

ST_MaxY

ST_MaxY は図形を入力パラメーターとし、その最大 Y 座標を戻します。

与えられた図形が 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:

```
▶▶ db2gse.ST_MaxY(—geometry—)▶▶
```

パラメーター:

geometry

最大 Y 座標を戻す、タイプ ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

DOUBLE

例:

次の例は、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 1:

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:

ST_MaxY

ID	MAX_Y
1	140
2	4
3	13

例 2:

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
```

関連概念:

- 437 ページの『ST_MaxX』

関連資料:

- 435 ページの『ST_MaxM』
- 440 ページの『ST_MaxZ』
- 450 ページの『ST_MinY』

ST_MaxZ

ST_MaxZ は図形を入力パラメーターとし、その最大 Z 座標を戻します。

与えられた図形が 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:

```
▶▶—db2gse.ST_MaxZ—(—geometry—)—————▶▶
```

パラメーター:

geometry

最大 Z 座標を戻す、タイプ ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

DOUBLE

例:

次の例は、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 1:

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 2:

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

関連概念:

- 437 ページの『ST_MaxX』

関連資料:

- 435 ページの『ST_MaxM』
- 439 ページの『ST_MaxY』
- 451 ページの『ST_MinZ』

ST_MBR

ST_MBR は図形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。

与えられた図形がポイントの場合は、ポイント自体が戻されます。図形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、図形の最小外接長方形がポリゴンとして戻されます。与えられた図形が 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:

▶ db2gse.ST_MBR(*geometry*)

パラメーター:*geometry*

外接する最小の長方形を戻す図形を表す、タイプが ST_Geometry (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、ポリゴンに外接する最小の長方形を戻す方法を示しています。指定された図形はポリゴンなので、図形に外接する最小の長方形はポリゴンとして戻されます。

次の例で、結果の線は読み易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))

関連資料:

- 391 ページの『ST_Envelope』
- 443 ページの『ST_MBRIntersects』

ST_MBRIntersects

ST_MBRIntersects は 2 つの図形を入力パラメーターとし、2 つの図形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、図形そのものです。

2 番目の図形が 1 番目の図形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。

与えられた図形のいずれかが NULL または空の場合は、NULL が戻されます。

構文:

```
►►—db2gse.ST_MBRIntersects—(—geometry1—,—geometry2—)—————►►
```

パラメーター:

geometry1

その図形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図形を表す、タイプが ST_Geometry またはそのサブタイプの 1 つの値。

geometry2

その図形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図形を表す値 (タイプ ST_Geometry (またはそのサブタイプの 1 つ))。

戻りタイプ:

INTEGER

例:

以下の例は、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) )
```

ST_MBRIntersects

例 1:

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	1 MBRs intersect
1	2	2 MBRs intersect
2	2	2 MBRs intersect
1	3	3 MBRs do not intersect
2	3	3 MBRs do not intersect
3	3	3 MBRs intersect

例 2:

次の SELECT ステートメントは、図形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

関連資料:

- 393 ページの『ST_EnvIntersects』
- 442 ページの『ST_MBR』

ST_MeasureBetween、ST_LocateBetween

ST_MeasureBetween または ST_LocateBetween は、図形および 2 つの M 座標 (指標) を入力パラメーターとし、その図形の、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。結果の図形は、与えられた図形の空間参照系で表現されます。

与えられた図形が面または複数面の場合、ST_MeasureBetween または ST_LocateBetween は図形の外部および内部リングに適用されます。与えられた図形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、空の図形が戻されます。与えられた図形が NULL の場合は NULL が戻されます。

ST_MeasureBetween および ST_LocateBetween

結果の図形は、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。

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:

```
db2gse.ST_MeasureBetween  
db2gse.ST_LocateBetween  
  
-(—geometry—,—startMeasure—,—endMeasure—)
```

パラメーター:

geometry

指標が *startMeasure* から *endMeasure* のものを見つける図形を表す、タイプが ST_Geometry 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タイプが DOUBLE 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タイプが DOUBLE 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。

図形の 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 (たとえば、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 をメジャー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。

ST_MeasureBetween および ST_LocateBetween

```
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:

```
►►—db2gse.ST_MidPoint—(—curve—)—————►►
```

パラメーター:

curve その中心を戻す曲線を表す、タイプが ST_Curve (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 は図形を入力パラメーターとし、その最小 M 座標を戻します。

与えられた図形が 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:

```
db2gse.ST_MinM(geometry)
```

パラメーター:

geometry

最小 M 座標を戻す、タイプ ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

DOUBLE

例:

次の例は、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,
```

ST_MinM

```
8 4 10 12,  
9 4 12 11,  
12 13 10 16))', 0) )
```

例 1:

この例は、SAMPLE_POLYS 内の各ポリゴンの最小の M 座標を見つけてみます。

```
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 2:

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 座標を見つけてみます。

```
SELECT CAST ( MIN ( ST_MinM(geometry) ) AS INTEGER) OVERALL_MIN_M  
FROM sample_polys
```

結果:

OVERALL_MIN_M
3

関連資料:

- 435 ページの『ST_MaxM』
- 448 ページの『ST_MinX』
- 450 ページの『ST_MinY』
- 451 ページの『ST_MinZ』

ST_MinX

ST_MinX は図形を入力パラメーターとし、その最小 X 座標を戻します。

与えられた図形が 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:

```
▶▶db2gse.ST_MinX(—geometry—)▶▶
```

パラメーター:

geometry

最小 X 座標を戻す、タイプ ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

DOUBLE

例:

次の例は、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 1:

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 2:

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

関連概念:

- 437 ページの『ST_MaxX』

関連資料:

- 447 ページの『ST_MinM』

ID	MIN_Y
1	120
2	0
3	4

例 2:

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
```

関連資料:

- 439 ページの『ST_MaxY』
- 447 ページの『ST_MinM』
- 448 ページの『ST_MinX』
- 451 ページの『ST_MinZ』

ST_MinZ

ST_MinZ は図形を入力パラメーターとし、その最小 Z 座標を戻します。

与えられた図形が 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:

```
▶▶—db2gse.ST_MinZ—(—geometry—)————▶▶
```

パラメーター:

geometry

最小 Z 座標を戻す、タイプ ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

DOUBLE

例:

次の例は、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)
```

ST_MinZ

```
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 1:

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 2:

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

関連資料:

- 440 ページの『ST_MaxZ』
- 447 ページの『ST_MinM』
- 448 ページの『ST_MinX』
- 450 ページの『ST_MinY』

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 を入力パラメータとして取り、対応する複数折れ線に戻します。

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:

```
db2gse.ST_MLineFromText ( wkt [, srs_id ] )
```

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、タイプが CLOB(2G) の値。

srs_id 結果の複数折れ線の空間参照系を識別する、タイプが 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_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、図形は空間参照系 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。この図形の 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) )
```

ST_MLineFromText

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 8.00000000, 43.00000000 12.00000000),
                      ( 39.00000000 3.00000000, 37.00000000 4.00000000,
                      36.00000000 7.00000000 ))
```

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 454 ページの『ST_MLineFromWKB』
- 462 ページの『ST_MultiLineString』

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 が戻されません。

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:

```
db2gse.ST_MLineFromWKB( ( wkb [, srs_id] ) )
```

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、タイプが BLOB(2G) の値。

srs_id 結果の複数折れ線の空間参照系を識別する、タイプが 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、例外条件が起きます (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。この図形は空間参照系 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。この図形の 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 8.00000000), (69.00000000 3.00000000, 67.00000000 4.00000000, 66.00000000 7.00000000, 68.00000000 9.00000000))

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 453 ページの『ST_MLineFromText』

- 462 ページの『ST_MultiLineString』
- 532 ページの『事前割り当てバイナリー (WKB) 表記』

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:

```
db2gse.ST_MPointFromText(wkt [, srs_id])
```

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、例外条件が起ります (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、図形は空間参照系 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。この図形の 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)

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 457 ページの『ST_MPointFromWKB』
- 464 ページの『ST_MultiPoint』
- 527 ページの『事前割り当てテキスト (WKT) 表記』

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 が戻されません。

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:

```
db2gse.ST_MPointFromWKB( ( wkb [, srs_id] ) )
```

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、例外条件が起こります (SQLSTATE 38SU1)。

戻りタイプ:

db2gse.ST_MultiPoint

例:

ST_MPointFromWKB

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。この図形は空間参照系 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。この図形の 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 列がありません。

```
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)
```

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 456 ページの『ST_MPointFromText』
- 464 ページの『ST_MultiPoint』
- 532 ページの『事前割り当てバイナリー (WKB) 表記』
- 477 ページの『ST_Point』

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:

```
db2gse.ST_MPolyFromText(—wkt—, —srs_id—)
```

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_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、図形は空間参照系 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。この図形の 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

```
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 28.00000000, 8.00000000 24.00000000)), (3.00000000 3.00000000, 5.00000000 3.00000000, 4.00000000 6.00000000, 3.00000000 3.00000000)))

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 460 ページの『ST_MPolyFromWKB』
- 465 ページの『ST_MultiPolygon』
- 527 ページの『事前割り当てテキスト (WKT) 表記』

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 が戻されません。

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:

```
db2gse.ST_MPolyFromWKB ( ( wkb , srs_id ) )
```

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。この図形は空間参照系 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。この図形の 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 79.00000000, 1.00000000
```

ST_MPolyFromWKB

```
72,00000000)),  
  ( 9.00000000 43.00000000, 6.00000000  
 47.00000000, 7.00000000 44.00000000, 9.00000000  
43.00000000 ))))
```

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 459 ページの『ST_MPolyFromText』
- 465 ページの『ST_MultiPolygon』
- 532 ページの『事前割り当てバイナリー (WKB) 表記』
- 487 ページの『ST_Polygon』

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
- ジオグラフィー・マークアップ言語 (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:

```
▶▶ db2gse.ST_MultiLineString ( ( wkt | wkb | gml | shape ) [, srs_id] ) ▶▶
```

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、タイプが CLOB(2G) 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、タイプが BLOB(2G) の値。
- gml* ジオグラフィー・マークアップ言語 (GML) を使用した、結果の複数折れ線を表す、タイプが CLOB(2G) の値。
- shape* 結果の複数折れ線の形状表記を表す、タイプが BLOB(2G) の値。
- srs_id* 結果の複数折れ線の空間参照系を識別する、タイプが 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、例外条件が起きます (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、図形は空間参照系 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。この図形の 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に戻します。

```
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 8.00000000, 43.00000000 12.00000000), (39.00000000 3.00000000, 37.00000000 4.00000000, 36.00000000 7.00000000))

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 527 ページの『事前割り当てテキスト (WKT) 表記』
- 532 ページの『事前割り当てバイナリー (WKB) 表記』
- 533 ページの『形状表記』
- 534 ページの『ジオグラフィー・マークアップ言語 (GML) 表記』

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
- ジオグラフィー・マークアップ言語 (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:

```

▶▶ db2gse.ST_MultiPoint ( ( wkt
                          | wkb
                          | gml
                          | shape
                          ) , -srs_id )

```

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 ジオグラフィー・マークアップ言語 (GML) を使用した、結果の複数ポイントを表す、タイプが CLOB(2G) の値。

shape 結果の複数ポイントの形状表記を表す、タイプが 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。

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、例外条件が起きます (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、図形は空間参照系 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。この図形の 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)
```

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 527 ページの『事前割り当てテキスト (WKT) 表記』
- 532 ページの『事前割り当てバイナリー (WKB) 表記』
- 533 ページの『形状表記』
- 534 ページの『ジオグラフィー・マークアップ言語 (GML) 表記』

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
- ジオグラフィー・マークアップ言語 (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:

```
►►—db2gse.ST_MultiPolygon—(wkt—wkb—shape—gml—)—————►
                               [,—srs_id—]
```

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。

ST_MultiPolygon

gml ジオグラフィー・マークアップ言語 (GML) を使用した、結果の複数ポリゴンを表す、タイプが CLOB(2G) の値。

shape 結果の複数ポリゴンの形状表記を表す、タイプが 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。

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、図形は空間参照系 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。この図形の 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 28.00000000, 8.00000000 24.00000000)),
      (( 3.00000000 3.00000000, 5.00000000 3.00000000,
4.00000000 6.00000000, 3.00000000 3.00000000)))
```

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 527 ページの『事前割り当てテキスト (WKT) 表記』
- 532 ページの『事前割り当てバイナリー (WKB) 表記』
- 533 ページの『形状表記』
- 534 ページの『ジオグラフィー・マークアップ言語 (GML) 表記』

ST_NumGeometries

ST_NumGeometries は図形集合を入力パラメーターとし、その集合内にある図形の数を返します。

与えられた図形集合が 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:

```
►—db2gse.ST_NumGeometries—(—collection—)—————►
```

パラメーター:

collection

図形の数を返す図形集合を表す、タイプが ST_GeomCollection (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

INTEGER

例:

2 つの図形集合を 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、それぞれの図形集合内に個々の図形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_NumGeometries

関連資料:

- 408 ページの『ST_GeometryN』

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:

```
db2gse.ST_NumInteriorRing(—polygon—)
```

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、表の中の図形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

関連資料:

- 415 ページの『ST_InteriorRingN』

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返します。

与えられた複数折れ線が NULL または空の場合は、NULL が返されます。

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:

```
►►—db2gse.ST_NumLineStrings—(—multilinestring—)—————►►
```

パラメーター:

multilinestring

折れ線の数を返す複数折れ線を表す、タイプが ST_MultiLineString 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図形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

関連資料:

- 433 ページの『ST_LineStringN』

ST_NumPoints

ST_NumPoints は、図形を入力パラメーターとし、その図形を定義するために使用されたポイントの数を返します。たとえば、図形がポリゴンであり、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、返される数は 5 です。

与えられた図形が NULL または空の場合は、NULL が返されます。

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:

```
►► db2gse.ST_NumPoints(—geometry—)◄◄
```

パラメーター:

geometry

ポイントの数を返す図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

INTEGER

例:

表に各種の図形を保管します。ST_NumPoints 関数により、SAMPLE_GEOMETRIES 表の各図形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

関連資料:

- 482 ページの『ST_PointN』

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返します。

与えられた複数ポリゴンが NULL または空の場合は、NULL が返されます。

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:

```
▶▶ db2gse.ST_NumPolygons(—multipolygon—)▶▶
```

パラメーター:

multipolygon

ポリゴンの数を返す複数ポリゴンを表す、タイプが ST_MultiPolygon 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図形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

関連資料:

- 489 ページの『ST_PolygonN』

ST_Overlaps

ST_Overlaps は 2 つの図形を入力パラメーターとし、その 2 つの図形の交差が同じディメンションの図形になるが、そのどちらとも等しくない図形になった場合に 1 を戻します。それ以外の場合、0 (ゼロ) が戻されます。

2 つの図形のいずれかが NULL または空の場合は、NULL が戻されます。

2 番目の図形が 1 番目の図形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:

```
►►db2gse.ST_Overlaps(—geometry1—,—geometry2—)—————►►
```

パラメーター:

geometry1

geometry2 とのオーバーラップをテストされる図形を表す、タイプが ST_Geometry (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図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

INTEGER

例:

次の例は、ST_Overlaps の使用法を示しています。各種の図形を作成し、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 50))', 1) )
```

例 1:

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 2:

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 3:

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

関連資料:

- 128 ページの『照会を最適化するために索引を使用する関数』

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:

```
db2gse.ST_Perimeter(surface [, unit])
```

パラメーター:

surface 周囲の長さを戻す、タイプが ST_Surface、ST_MultiSurface、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* が投影座標系、または地心から見た座標系の場合、この座標系に関連付けられた線形単位がデフォルトになります。
- *surface* が地理座標系で、測地地理座標系 (SRS) でない場合、この座標系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 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。

- 図形の座標系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。
- 図形の座標系が投影座標系で、かつ角度単位が指定されている。
- 図形の座標系が地理座標系で、測地 SRS でなく、かつ線形単位が指定されている。
- 図形の座標系が地理座標系かつ測地 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:

以下の例は、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 の図形を入れます。

```
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 1:

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	+1.8000000000000000E+001

例 2:

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:

ST_PerpPoints

▶▶ db2gse.ST_PerpPoints(—curve—,—point—)▶▶

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、タイプが ST_Curve または ST_MultiCurve (またはこれらのサブタイプ) の値。

point *curve* 上に垂直に投影するポイントを表す、タイプが ST_Point の値。

戻りタイプ:

db2gse.ST_MultiPoint

例:

以下の例は、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見つけます。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 1:

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 2:

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 VARCHAR(160) ) PERP
FROM sample_lines
```

結果:

```
PERP
-----
MULTIPOINT ( 0.00000000 5.00000000, 5.00000000 0.00000000, 10.00000000 5.00000000)
```

例 3:

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 4:

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 5:

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、空の図形が戻されます。

```
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
- ジオグラフィー・マークアップ言語 (GML) 表記

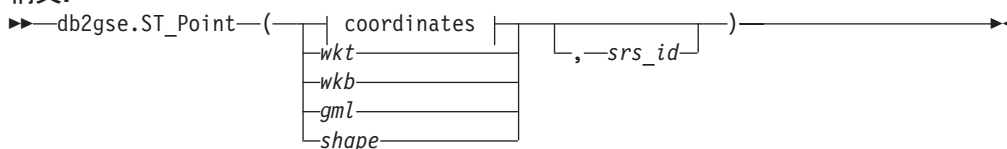
結果のポイントを置く空間参照系を示すため、オプションとして空間参照系 ID を指定することができます。

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

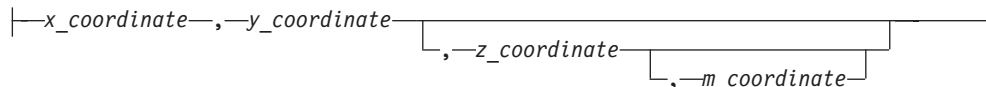
ST_Point

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:



coordinates:



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、タイプが BLOB(2G) の値。
- gml* ジオグラフィック・マークアップ言語 (GML) を使用した、結果のポイントを表す、タイプが CLOB(2G) の値。
- shape* 結果のポイントの形状表記を表す、タイプが BLOB(2G) の値。
- srs_id* 結果のポイントの空間参照系を識別する、タイプが 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)。
- x_coordinate* 結果のポイントの X 座標を示す、タイプが DOUBLE の値。
- y_coordinate* 結果のポイントの Y 座標を示す、タイプが DOUBLE の値。
- z_coordinate* 結果のポイントの Z 座標を示す、タイプが DOUBLE 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、タイプが DOUBLE 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 の図形です。

```
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に戻します。

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(90)) POINTS
FROM sample_points
```

結果:

ID	POINTS
1100	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 (120.00000000 358.00000000 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、表記としてジオグラフィー・マークアップ言語を使用して、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
```

ST_Point

結果:

```
ID          POINTS
-----
1104 POINT Z ( 1003.000000 9876.000000 20.00000000)
```

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 419 ページの『ST_Is3d』
- 423 ページの『ST_IsMeasured』
- 527 ページの『事前割り当てテキスト (WKT) 表記』
- 532 ページの『事前割り当てバイナリー (WKB) 表記』
- 533 ページの『形状表記』
- 534 ページの『ジオグラフィー・マークアップ言語 (GML) 表記』

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:

```
▶▶ db2gse.ST_PointFromText ( ( wkt [, srs_id ] ) )
```

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、タイプが 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、例外条件が起きます (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、

図形は空間参照系 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。この図形の 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)
```

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 477 ページの『ST_Point』
- 481 ページの『ST_PointFromWKB』

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:

```
db2gse.ST_PointFromWKB( (wkb) [, (srs_id)] )
```

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、タイプが BLOB(2G) の値。

srs_id 結果のポイントの空間参照系を識別する、タイプが 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、例外条件が起こります (SQLSTATE 38SU1)。

ST_PointFromWKB

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。図形は空間参照系 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 列からポイントに戻します。

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)

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 477 ページの『ST_Point』
- 480 ページの『ST_PointFromText』

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に戻します。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:

▶▶ db2gse.ST_PointN (*geometry* , *index*) ▶▶

パラメーター:

geometry

index で指定されたポイントに戻す図形を表す、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、タイプが INTEGER 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 (5.000000000 5.000000000)

関連資料:

- 390 ページの『ST_Endpoint』
- 470 ページの『ST_NumPoints』
- 495 ページの『ST_StartPoint』

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に戻します。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:

ST_PointOnSurface

▶▶ db2gse.ST_PointOnSurface (—*surface*—) ▶▶

パラメーター:

surface 面上のポイントを戻す図形を表す、タイプが ST_Surface、ST_MultiSurface、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:

▶▶ db2gse.ST_PolyFromText (—*wkt*—
 └─*srs_id*—) ▶▶

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、タイプが CLOB(2G) の値。
- srs_id* 結果のポリゴンの空間参照系を識別する、タイプが 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_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、図形は空間参照系 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。この図形の 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))
```

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 485 ページの『ST_PolyFromWKB』
- 487 ページの『ST_Polygon』

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。

ST_PolyFromWKB

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:

```
db2gse.ST_PolyFromWKB(wkb [, srs_id])
```

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、タイプが BLOB(2G)
の値。

srs_id 結果のポリゴンの空間参照系を識別する、タイプが 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_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。この図形は空間参照系 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
ゴンに戻します。この図形の 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))
```

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 484 ページの『ST_PolyFromText』
- 487 ページの『ST_Polygon』

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
- ジオグラフィー・マークアップ言語 (GML) 表記

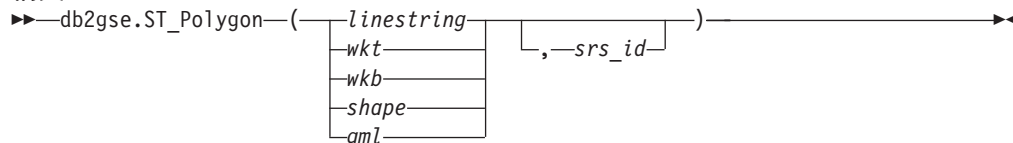
結果のポリゴンを入れる空間参照系を示すために、オプションとして空間参照系 ID を指定することができます。

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、タイプが ST_LineString の値。*linestring* が閉じておらず、かつ単純な場合、例外条件が起きます (SQLSTATE 38SSSL)。

ST_Polygon

- wkt* 結果のポリゴンの事前割り当てテキスト表記が入る、タイプが CLOB(2G) 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、タイプが BLOB(2G) の値。
- shape* 結果のポリゴンの形状表記を表す、タイプが BLOB(2G) の値。
- gml* ジオグラフィー・マークアップ言語 (GML) を使用した、結果のポリゴンを表す、タイプが CLOB(2G) の値。
- srs_id* 結果のポリゴンの空間参照系を識別する、タイプが INTEGER 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、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 の図形です。

- 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, 125) (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に戻します。

```
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))

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 527 ページの『事前割り当てテキスト (WKT) 表記』
- 532 ページの『事前割り当てバイナリー (WKB) 表記』
- 533 ページの『形状表記』
- 534 ページの『ジオグラフィー・マークアップ言語 (GML) 表記』

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に戻します。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:

```
►►—db2gse.ST_PolygonN—(—multipolygon—,—index—)—————►►
```

パラメーター:

multipolygon

index で指定されたポリゴンに戻す複数ポリゴンを表す、値 (タイプ ST_MultiPolygon)。

ST_PolygonN

index *multipolygon* から戻される *n* 番目のポリゴンを示す、タイプ INTEGER 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))

関連資料:

- 471 ページの『ST_NumPolygons』

ST_Relate

ST_Relate は、2 つの図形と DE-9IM (Dimensionally Extended 9 Intersection Model) マトリックスを入力パラメーターとし、与えられた図形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。

与えられた図形のいずれかが NULL 値または空の場合は、NULL 値が戻されません。

2 番目の図形が 1 番目の図形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:

```
▶▶—db2gse.ST_Relate—(—geometry1—,—geometry2—,—matrix—)————▶▶
```

パラメーター:

geometry1

geometry2 に対してテストされる図形を表す、タイプが ST_Geometry (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図形を表す、タイプが ST_Geometry (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。たとえば、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***FF2') "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

関連資料:

- 318 ページの『地勢を比較する関数』

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。結果の図形は、与えられた図形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。

ST_RemovePoint

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_RemovePoint(curve, point)
```

パラメーター:

curve *point* を除去する曲線を表す、タイプが ST_Curve (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、タイプ ST_Point の値。

戻りタイプ:

db2gse.ST_Curve

例:

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 1:

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 2:

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、図形および (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、図形の空間参照系を指定された空間参照系に変更した図形が戻されます。図形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、与えられた図形の現行の空間参照系 ID が戻されます。

与えられた図形が 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:

```
►► db2gse.ST_SrsId (—geometry— [—srs_id—])
```

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 を戻す図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

srs_id 結果の図形に使用される空間参照系を識別する、タイプが INTEGER の値。

重要: このパラメーターを指定した場合、図形はトランスフォームされませんが、図形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。

ST_SrsId および ST_SRID

```
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

関連資料:

- 507 ページの『ST_Transform』

ST_SrsName

ST_SrsName は図形を入力パラメーターとし、与えられた図形を表現する空間参照系の名前を戻します。

与えられた図形が 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:

▶—db2gse.ST_SrsName—(—*geometry*—)————▶

パラメーター:

geometry

空間参照系の名前を戻す図形を表す、タイプが ST_Geometry (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てみます。

```
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

関連資料:

- 493 ページの『ST_SrsId、ST_SRID』

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:

```
►►—db2gse.ST_StartPoint—(—curve—)—————▶▶
```

パラメーター:

curve 最初のポイントを戻す図形を表す、タイプが ST_Curve (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:

ST_StartPoint

```
ID          START_POINT
-----
1 POINT ( 10.00000000 10.00000000)
2 POINT Z ( 0.00000000 0.00000000 4.00000000)
```

関連資料:

- 390 ページの『ST_Endpoint』
- 482 ページの『ST_PointN』

ST_SymDifference

ST_SymDifference は 2 つの図形を入力パラメーターとし、与えられた 2 つの図形の対称差である図形を戻します。対称差 (symmetrical difference) とは、与えられた 2 つの図形の交差していない部分です。結果の図形は、1 番目の図形の空間参照系で表現されます。戻される図形のディメンションは、入力された図形と同じになります。2 つの図形は、同じディメンションの図形でなければなりません。

非測地データの場合、2 番目の図形が 1 番目の図形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、両方の図形が同じ測地参照系 (SRS) で表現される必要があります。

図形が等しい場合、タイプが ST_Point の空の図形が戻されます。いずれかの図形が NULL の場合は NULL が戻されます。

結果の図形は、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:

```
▶▶—db2gse.ST_SymDifference—(—geometry1—,—geometry2—)————▶▶
```

パラメーター:

geometry1

geometry2 との対称差を計算する、1 番目の図形を表す、タイプが ST_Geometry (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 番目の図形を表す、タイプが ST_Geometry (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

- 両方の図形が測地で、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:

次の例は、ST_SymDifference 関数の使用法を示します。図形を 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 1:

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 2:

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:

ST_SymDifference

```
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 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))
```

関連資料:

- 379 ページの『ST_Difference』

ST_ToGeomColl

ST_ToGeomColl は図形を入力パラメーターとし、これを図形の集合に変換します。結果である図形の集合は、与えられた図形の空間参照系で表現されます。

指定された図形が空の場合、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。

指定された図形が空でない場合は、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。

与えられた図形が 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:

```
▶▶ db2gse.ST_ToGeomColl(—geometry—)▶▶
```

パラメーター:

geometry

図形の集合に変換される図形を表す、タイプが ST_Geometry (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、図形をその対応する図形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 は図形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、与えられた図形の空間参照系で表現されます。

与える図形は空または折れ線である必要があります。与えられた図形が 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:

```
►►—db2gse.ST_ToLineString—(—geometry—)—————►►
```

パラメーター:*geometry*

折れ線に変換される図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

図形は、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:

ST_ToLineString

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に戻します。

```
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 は図形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、与えられた図形の空間参照系で表現されます。

与える図形は空、複数折れ線、または折れ線である必要があります。与えられた図形が 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:

```
▶▶—db2gse.ST_ToMultiLine—(—geometry—)————▶▶
```

パラメーター:

geometry

複数折れ線に変換される図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

図形が、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に戻します。

```
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 は図形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、与えられた図形の空間参照系で表現されます。

与える図形は空、ポイント、または複数ポイントである必要があります。与えられた図形が 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:

```
►►—db2gse.ST_ToMultiPoint—(—geometry—)—————►►
```

パラメーター:

geometry

複数ポイントに変換される図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

図形は、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)。

ST_ToMultiPoint

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 は図形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、与えられた図形の空間参照系で表現されます。

与える図形は空、ポリゴン、または複数ポリゴンである必要があります。与えられた図形が 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:

```
▶▶—db2gse.ST_ToMultiPolygon—(—geometry—)————▶▶
```

パラメーター:

geometry

複数ポリゴンに変換される図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

図形は、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。

この例は複数の図形を作成し、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 は図形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、与えられた図形の空間参照系で表現されます。

与える図形は空またはポイントである必要があります。与えられた図形が 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:

```
►► db2gse.ST_ToPoint (—geometry—) ◀◀
```

パラメーター:

geometry

ポイントに変換される図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

ST_ToPoint

図形は、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 つの図形を作成し、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に戻します。

```
SELECT CAST( ST_AsText( ST_ToPoint(geometry) ) AS VARCHAR(35) ) POINTS
FROM sample_geometries
```

結果:

```
POINTS
-----
POINT ( 30.000000000 40.000000000)
POINT EMPTY
POINT EMPTY
```

ST_ToPolygon

ST_ToPolygon は図形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、与えられた図形の空間参照系で表現されます。

与える図形は空またはポリゴンである必要があります。与えられた図形が 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:

```
▶▶ db2gse.ST_ToPolygon(—geometry—)▶▶
```

パラメーター:

geometry

ポリゴンに変換される図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

図形は、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 つの図形を作成し、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に戻します。

```
SELECT CAST( ST_AsText( ST_ToPolygon(geometry) ) AS VARCHAR(130) ) POLYGONS
FROM sample_geometries
```

結果:

POLYGONS

```
-----
POLYGON (( 0.000000000 0.000000000, 5.000000000 0.000000000,
           5.000000000 4.000000000,0.000000000 4.000000000,
           0.000000000 0.000000000))
```

POLYGON EMPTY

POLYGON EMPTY

ST_Touches

ST_Touches は 2 つの図形を入力パラメーターとし、与えられた図形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 つの図形が接触する」とは、両方の図形の内部は交差していないが、一方の図形の境界が、他方の図形の境界または内部と交差している場合です。

2 番目の図形が 1 番目の図形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。

与えられた図形の両方がポイントまたは複数ポイントの場合、または与えられた図形のいずれかが NULL または空の場合は、NULL が戻されます。

構文:

```
▶▶—db2gse.ST_Touches—(—geometry1—,—geometry2—)————▶▶
```

パラメーター:

ST_Touches

geometry1

geometry2 との接触をテストする図形を表す、タイプが ST_Geometry (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図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

戻りタイプ:

INTEGER

例:

いくつかの図形を SAMPLE_GEOMS 表に追加します。次に ST_Touches 関数を使用して、どの図形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

関連資料:

- 128 ページの『照会を最適化するために索引を使用する関数』

ST_Transform

ST_Transform は、図形および空間参照系 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図形をトランスフォームします。異なる座標系の間で投影および変換が行われ、図形の座標はそれに従って調整されます。

図形を指定された空間参照系に変換できるのは、図形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系に基づく場合のみです。図形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、投影座標系に基づく場合、投影されたものの基礎にある地理座標系を判別するため、逆の投影が行われます。

与えられた図形が 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:

```
▶▶ db2gse.ST_Transform(—geometry—, —srs_id—)▶▶
```

パラメーター:

geometry

srs_id で示された空間参照系に変換される図形を表す、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

srs_id 結果の図形の空間参照系を識別する、タイプが INTEGER 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:

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図形を変換しています。

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 座標系の 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。

ST_Transform

```
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 関数を使用して図形を変換します。

例 1:

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 2:

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)

関連資料:

- 305 ページの『DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビュー』

ST_Union

ST_Union は 2 つの図形を入力パラメーターとし、与えられた図形の和である図形を戻します。結果の図形は、1 番目の図形の空間参照系で表現されます。

2 つの図形は、同じディメンションの図形でなければなりません。与えられた 2 つの図形のいずれかが NULL の場合は、NULL が戻されます。

非測地データの場合、2 番目の図形が 1 番目の図形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、両方の図形が同じ測地参照系 (SRS) で表現される必要があります。

結果の図形は、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:

```
▶▶ db2gse.ST_Union(—geometry1—, —geometry2—) ▶▶
```

パラメーター:

geometry1

geometry2 と結合される、タイプが ST_Geometry (またはそのサブタイプの 1 つ) の値。

geometry2

geometry1 と結合される図形を表す、タイプが ST_Geometry (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 : 両方の図形が測地で、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:

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。

ST_Union

例 1:

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 2:

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 3:

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 つの図形を入力パラメーターとし、1 番目の図形が完全に 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。

与えられた図形のいずれかが NULL 値または空の場合は、NULL 値が戻されます。

ST_Within

```
POINT_ID_WITHIN_POLYGONS
-----
2
```

例 2:

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
```

関連資料:

- 373 ページの『ST_Contains』

ST_WKBToSQL

ST_WKBToSQL は、図形の事前割り当てバイナリー表記を取り、これに対応する図形を戻します。結果の図形には、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 が戻されません。

ST_WKBToSQL(*wkb*) は、ST_Geometry(*wkb*,0) と同じ結果が得られます。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:

```
▶▶ db2gse.ST_WKBToSQL(—wkb—)—————▶▶
```

パラメーター:

wkb 結果の図形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、図形を 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、この図形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_WKBTtoSQL 関数を使用して、WKB 列の図形の座標を戻します。

```
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 列にある図形を調べます。

```
SELECT id, CAST( ST_AsText( ST_WKBTtoSQL(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))

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 406 ページの『ST_Geometry』
- 513 ページの『ST_WKTTtoSQL』

ST_WKTTtoSQL

ST_WKTTtoSQL は、図形の事前割り当てテキスト表記を取り、対応する図形を戻します。結果の図形には、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_WKTTtoSQL(*wkt*) は、ST_Geometry(*wkt*,0) と同じ結果が得られます。

ST_WKTTtoSQL 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:

```
▶▶—db2gse.ST_WKTTtoSQL—(—wkt—)————▶▶
```

パラメーター:

wkt 結果の図形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:

ST_WKTTToSQL

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_WKTTToSQL により、図形の事前割り当てテキスト表記を使用して図形を作成、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_WKTTToSQL( 'point (44 14)' ) ),
      (11, ST_WKTTSQL ( 'point (24 13)' ) ),
      (12, ST_WKTTToSQL ('polygon ((50 20, 50 40, 70 30, 50 20))' ) )
```

この SELECT ステートメントは、挿入された図形を戻します。

```
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))

関連概念:

- 4 ページの『空間データと測地データ』

関連資料:

- 406 ページの『ST_Geometry』
- 512 ページの『ST_WKBTToSQL』

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:

```
db2gse.ST_X(point, x_coordinate)
```

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 座標を表す値 (タイプは DOUBLE)。

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:

次の例は、ST_X 関数の使用法を示しています。図形を作成し、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 1:

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	+2.000000000000000E+000
2	+4.000000000000000E+000
3	+3.000000000000000E+000

例 2:

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)

関連資料:

- 434 ページの『ST_M』
- 515 ページの『ST_Y』
- 517 ページの『ST_Z』

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

ST_Y

- 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:

```
db2gse.ST_Y(point, y_coordinate)
```

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 座標を表す値 (タイプは DOUBLE)。

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:

次の例は、ST_Y 関数の使用法を示しています。図形を作成し、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 1:

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	+3.0000000000000000E+000
2	+5.0000000000000000E+000
3	+8.0000000000000000E+000

例 2:

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)**関連資料:**

- 434 ページの『ST_M』
- 514 ページの『ST_X』
- 517 ページの『ST_Z』

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:

```
db2gse.ST_Z(point, z_coordinate)
```

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 の新しい Z 座標を表す値 (タイプは DOUBLE)。

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:

次の例は、ST_Z 関数の使用法を示しています。図形を作成し、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 1:

この例は、表内のポイントの Z 座標を見つけます。

ST_Z

```
SELECT id, ST_Z (geometry) Z_COORD
FROM sample_points
```

結果:

```
ID          Z_COORD
-----
1 +3.200000000000000E+001
2 +2.000000000000000E+001
3 +2.300000000000000E+001
```

例 2:

この例は、Z 座標が 40 のポイントに戻します。

```
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)
```

関連資料:

- 434 ページの『ST_M』
- 514 ページの『ST_X』
- 515 ページの『ST_Y』

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、表内の図形の列を和集約として集約し、1 つの図形にします。

和として結合する図形のすべてが NULL の場合は、NULL が戻されます。和として結合する図形のそれぞれが NULL または空の場合は、タイプが ST_Point の空の図形が戻されます。

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:

```
▶▶—db2gse.ST_GetAggrResult—(—————)—————▶
▶—MAX—(—db2sge.ST_BuildUnionAggr—(—geometries—)—)—)—————▶▶
```

パラメーター:

geometries

タイプが ST_Geometry (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図形を表す列。

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、表の地理情報列の和集約を作成できません。

- MPP (超並列プロセッサ) 環境内
- SELECT で GROUP BY 文節を使用した場合
- 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図形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)

地理情報列を含んでいる表からデータを検索する場合、地理情報列からのデータは、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か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。

関連概念:

- 521 ページの『ST_WellKnownText トランスフォーム・グループ』
- 523 ページの『ST_WellKnownBinary トランスフォーム・グループ』
- 524 ページの『ST_Shape トランスフォーム・グループ』
- 525 ページの『ST_GML 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

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図形を WKT 表記に変換するために使用されます。図形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

ST_WellKnownText

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、図形は、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_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図形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 ( 100.000000000 100.000000000, 200.000000000 100.000000000)
```

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 関数を使用して図形を挿入するために、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);
```


次の C コードは、タイプ BLOB の変数で、挿入される図形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 関数を使用して図形を挿入するために、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、図形をその 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、図形は、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、図形をその 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
```

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 関数を使用せずに図形を挿入するために、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

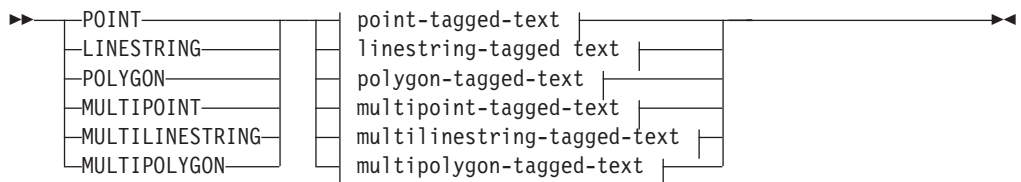
この章では、DB2 Spatial Extender で使用できる、業界標準の空間データ・フォーマットを説明します。これらのフォーマットを受け付け、作成する関数については、309 ページの『図形値をデータ交換フォーマットに変換する空間処理関数』を参照してください。これらのフォーマットを含むファイルのインポートおよびエクスポートについては、89 ページの『空間データのインポートとエクスポートについて』を参照してください。以下の 4 つの空間データ・フォーマットが説明されています。

- 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
- ジオグラフィー・マークアップ言語 (GML) 表記

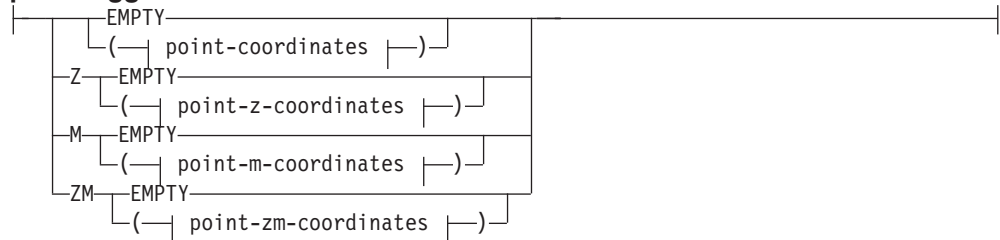
事前割り当てテキスト (WKT) 表記

OpenGIS コンソーシアムの「Simple Features for SQL」仕様に、ASCII フォーマットの図形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図形を変換する空間処理関数』を参照してください。

図形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。

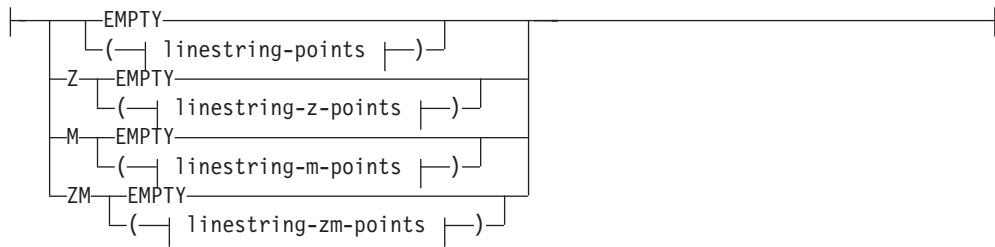


point-tagged-text:

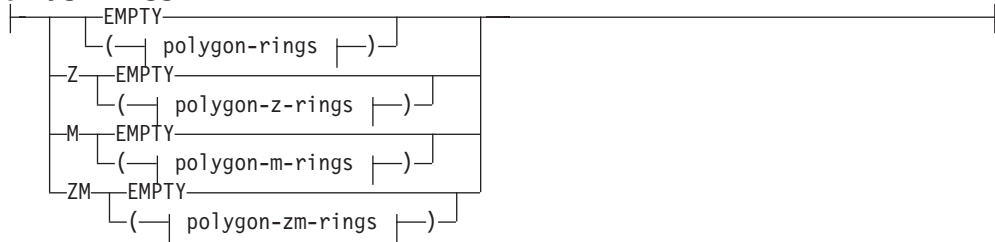


linestring-tagged-text:

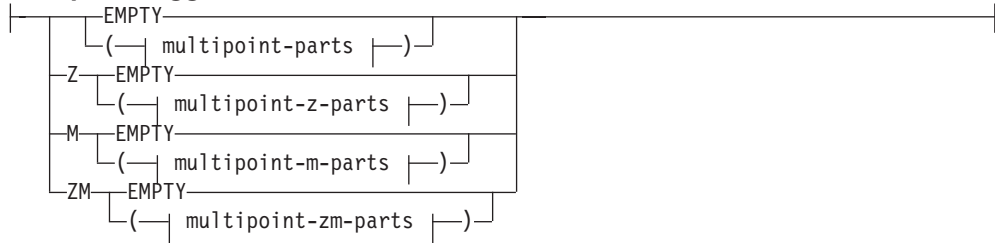
事前割り当てテキスト (WKT) 表記



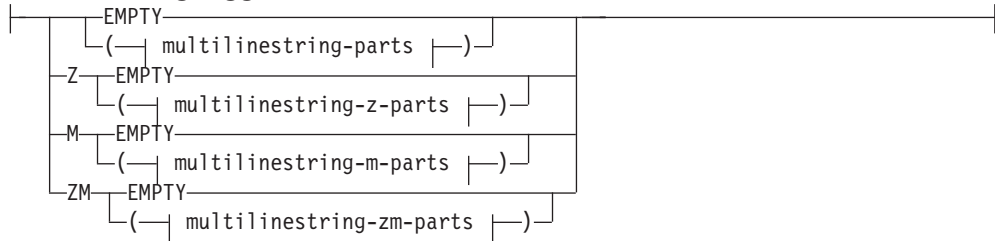
polygon-tagged-text:



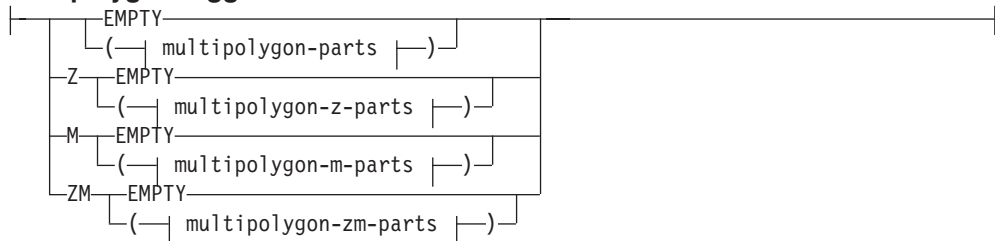
multipoint-tagged-text:



multilinestring-tagged-text:



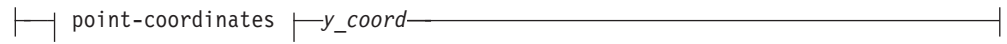
multipolygon-tagged-text:



point-coordinates:



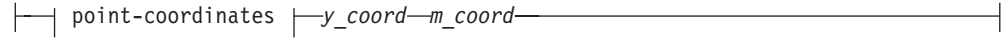
point-z-coordinates:



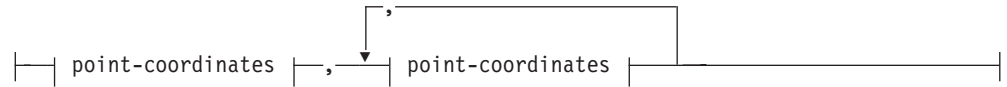
point-m-coordinates:



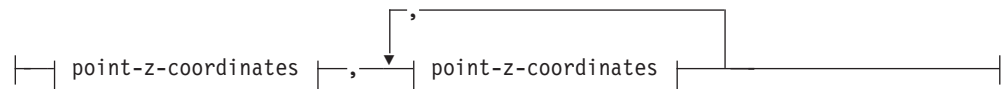
point-zm-coordinates:



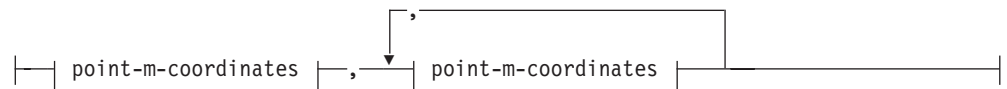
linestring-points:



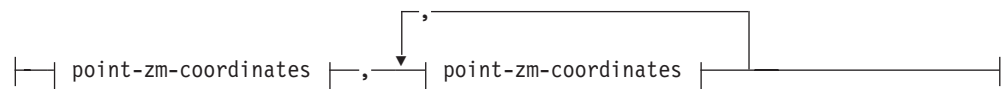
linestring-z-points:



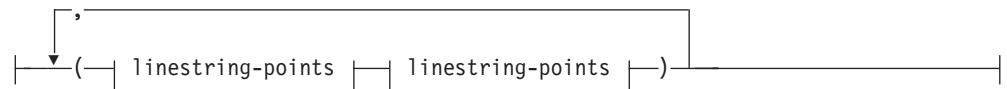
linestring-m-points:



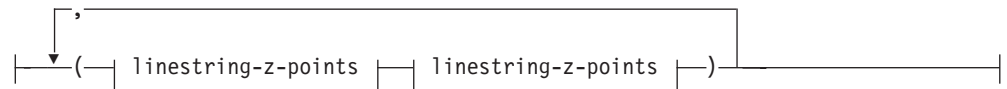
linestring-zm-points:



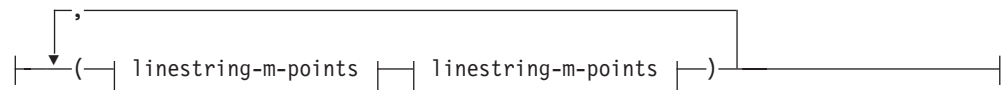
polygon-rings:



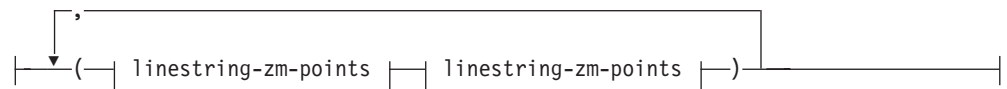
polygon-z-rings:



polygon-m-rings:



polygon-zm-rings:



事前割り当てテキスト (WKT) 表記

multipoint-parts:



multipoint-z-parts:



multipoint-m-parts:



multipoint-zm-parts:



multilinestring-parts:



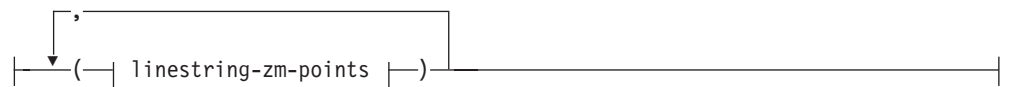
multilinestring-z-parts:



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)。

図形が空の場合は、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しないでください。

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 57. 図形のタイプとそのテキスト表記

図形タイプ	WKT 表記	コメント
ポイント	POINT EMPTY	空のポイント
ポイント	POINT (10.05 10.28)	ポイント
ポイント	POINT Z(10.05 10.28 2.51)	Z 座標を持つポイント
ポイント	POINT M(10.05 10.28 4.72)	M 座標を持つポイント
ポイント	POINT ZM(10.05 10.28 2.51 4.72)	Z 座標と M 座標を持つポイント
折れ線	LINestring EMPTY	空の折れ線
ポリゴン	POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))	ポリゴン
複数ポイント	MULTIPOINT Z(10 10 2, 20 20 3)	Z 座標を持つ複数ポイント
複数折れ線	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
複数ポリゴン	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

関連資料:

- 309 ページの『図形値をデータ交換フォーマットに変換する空間処理関数』

事前割り当てバイナリー (WKB) 表記

このセクションでは、図形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。他の図形のバイト・ストリームは、既に定義済みの図形のバイト・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図形の例を示しています。



図 58. NDR フォーマットの図形表記：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)。

関連資料:

- 309 ページの『図形値をデータ交換フォーマットに変換する空間処理関数』

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 ウェブ・サイト (<http://www.esri.com/library/whitepapers/pdfs/shapfile.pdf>) を参照してください。

下記の関連参照セクションの『図形とデータ交換フォーマットの間の変換をする空間処理関数』トピックに、形状データ・フォーマットを受け付け、作成する、空間処理関数が説明されています。

関連資料:

- 309 ページの『図形値をデータ交換フォーマットに変換する空間処理関数』

ジオグラフィー・マークアップ言語 (GML) 表記

DB2 Spatial Extender には、ジオグラフィー・マークアップ言語 (GML) 表記から図形を生成するいくつかの関数があります。

下記の関連参照セクションの『データ交換フォーマットの図形を変換する空間処理関数』に、DB2 Spatial Extender が提供する、図形の値と GML 表記との変換を行う関数の詳細が説明されています。

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> にあります。

関連資料:

- 309 ページの『図形値をデータ交換フォーマットに変換する空間処理関数』

第 26 章 サポートされる座標系

この章では、空間データを解釈するために使用される座標値についての参照情報を記載しています。以下のトピックを取り上げます。

- 座標系の概要
- サポートされる線形単位
- サポートされる角度単位
- サポートされる回転楕円面
- サポートされる測地データ
- サポートされる本初子午線
- サポートされるマップ投影

サポートされる座標系

このトピックでは、DB2 Spatial Extender によりサポートされる座標系の構文を説明し、座標系の値をリストしています。

座標系の構文

空間参照系の事前割り当てテキスト表記は、座標系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。

座標系は、地理座標系 (緯度 - 経度) であるか、投影座標系 (X,Y) であるか、または地球の中心から見た (geocentric) 座標系 (X,Y,Z) です。座標系は、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 (たとえば、DATUM や UNIT など) を持ち、その後続けて、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系のストリング表記の EBNF (拡張バックス正規形式) 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>
```

座標系のタイプは、使用されるキーワードで示されます。

サポートされる座標系

PROJCS

データが投影座標にある場合は、データ・セットの座標系は 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・セットの座標系は 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・セットの座標系は 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。投影座標系の名前の後には、いくつかのオブジェクト (地理座標系、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。投影座標系はすべて、地理座標系に基づいているので、このセクションは、投影座標系に特有な項目を最初に説明します。たとえば、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]]
```

代わりに、名前および複数のオブジェクトが、地理座標系オブジェクト (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 の地理座標系ストリング:

```
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]]
```

地心から見た座標系は、地理座標系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 58. 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 59. 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
Radian	-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
Decimal Degree	-90 度以上、90 度以 下	-180 度以上、180 度 以下	pi/180
Decimal Minute	-5400 分以上、5400 分以下	-10800 分以上、10800 分以下	(pi/180)/60
Decimal Second	-324000 秒以上、 324000 秒以下	-648000 秒以上、 648000 秒以下	(pi/180)*3600

サポートされる座標系

表 59. 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
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
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 60. 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
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

表 60. サポートされる回転楕円面 (続き)

名前	半長軸	逆平坦化
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
球面	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 61. 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

サポートされる座標系

表 61. サポートされる測地データ (続き)

名前	測地系
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
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

表 61. サポートされる測地データ (続き)

名前	測地系
European Datum 1987	Qatar 1948
Egypt 1907	Qornoq
European Reference System 1989	RT38
Fahud	South American Datum 1969
Gandajika 1970	Sapper Hill 1943
Garoua	Schwarzeck
Geocentric Datum of Australia 1994	Segora
Guyane Francaise	Serindung
Herat North	Stockholm 1938
Hito XVIII 1963	Sudan
Hu Tzu	Shan Tananarive 1925
Hungarian Datum 1972	Timbalai 1948
Indian 1954	TM65
Indian 1975	TM75
Indonesian Datum 1974	Tokyo
Jamaica 1875	Trinidad 1903
Jamaica 1969	Trucial Coast 1948
Kalianpur	Voirol 1875
Kandawala	Voirol Unifie 1960
Kertau	WGS 1972
Kuwait Oil Company	WGS 1972 Transit Broadcast Ephemeris
La Canoa	WGS 1984
Lake	Yacare
Leigon	Yoff
Liberia 1964	Zanderij

サポートされる本初子午線

表 62. 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
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

サポートされる座標系

表 62. サポートされる本初子午線 (続き)

ロケーション	座標
Stockholm	18° 3' 29" E

サポートされるマップ投影

表 63. 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 64. コニック投影

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
Polyconic	Alaska series E
Simple conic	Alaska Grid (Modified-Stereographic by Snyder)

表 65. 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。

表 65. マップ投影パラメーター (続き)

パラメーター	説明
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。
perspective_point_height	マップ投影のパーспекティブ・ポイントの地上からの高さ。
fipszone	State Plane 座標系ゾーン番号。
zone	UTM ゾーン番号。

付録 A. 使用すべきでないストアード・プロシージャ

このトピックでは、使用すべきでないストアード・プロシージャの概要を説明しています。

注: 新しいアプリケーションはすべて、DB2 Spatial Extender バージョン 8 に定義されているストアード・プロシージャを使用して作成し、現行アプリケーションはバージョン 8 で定義されたストアード・プロシージャを使用するように更新してください。

使用すべきでないストアード・プロシージャは、以下の表に要約されている作業を実行していました。

表 66. 使用すべきでないストアード・プロシージャ

ストアード・プロシージャ名	ストアード・プロシージャ・タスク
db2gse.gse_enable_autogc	ジオコーダーが、空間列を対応する属性列に自動的に同期させることができるようにする。
db2gse.gse_enable_db	データベースで地理情報操作をサポート可能にする。
db2gse.gse_enable_idx	地理情報列に索引を作成する
db2gse.gse_enable_sref	空間参照系を作成する
db2gse.gse_export_shape	レイヤーおよびこれに関連付けられた表を形状ファイルにエクスポートする
db2gse.gse_disable_autogc	空間列を、対応する属性列に自動的に同期させることができないようにジオコーダーを使用不可にする
db2gse.gse_disable_db	データベースでの地理情報操作のサポートを使用不可にする
db2gse.gse_disable_sref	空間参照系をドロップする
db2gse.gse_import_shape	レイヤーおよびこれに関連付けられた表を ESRI_SDE 転送ファイルからインポートする
db2gse.gse_register_gc	デフォルト・ジオコーダー以外のジオコーダーを登録する
db2gse.gse_register_layer	地理情報列をレイヤーとして登録する
db2gse.gse_run_gc	ジオコーダーをバッチ・モードで実行する
db2gse.gse_unregist_gc	デフォルト・ジオコーダー以外のジオコーダーの登録を抹消する
db2gse.gse_unregist_layer	レイヤーの登録を抹消する

db2gse.gse_enable_autogc

このストアード・プロシージャは、以下の作業に使用します。

使用すべきでないストアード・プロシージャー

- 地理情報列をこれに関連する属性列 (複数可) と同期させるためのトリガーを作成する。属性列に値が挿入、または更新されるたびに、トリガーは登録されたジオコーダーを呼び出し、挿入または更新された値をジオコーディングし、結果のデータを地理情報列に入れます。
- トリガーが一時的に使用不可になった後、トリガーを再活動化します。
- 挿入および更新された値をジオコーディングするために使用する関数を設定します。

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、SYSADM 権限または DBADM 権限。
- この表に対する CONTROL 特権。
- この表に対する ALTER 特権、SELECT 特権、または UPDATE 特権。

パラメーター:

表 67. db2gse.gse_enable_autogc ストアード・プロシージャーの入力パラメーター

名前	データ・タイプ	説明
operMode	SMALLINT	<p>ジオコーディングを開始するトリガーが初めて作成されるのか、または一時的に使用不可にされた後で再活動化されるのかを示す値。</p> <p>このパラメーターは NULL にはできません。</p> <p>コメント: トリガーを作成するには、GSE_AUTOGC_CREATE マクロを使用します。トリガーを再活動化するには、GSE_AUTOGC_RECREATE マクロを使用します。これらのマクロに関連する値については、db2gse.h ファイルを参照してください。AIX では、このファイルは \$DB2INSTANCE/sqlib/include/ ディレクトリーにあります。Windows NT では、これは %DB2PATH%\include¥ ディレクトリーにあります。</p> <p>operMode パラメーターが GSE_AUTOGC_CREATE に設定されている場合、登録されたジオコーダーの ID を gcId パラメーターに指定する必要があります。</p>
layerSchema	VARCHAR(30)	<p>layerTable パラメーターに指定された表が属するスキーマの名前。</p> <p>このパラメーターは NULL にできます。</p> <p>layerSchema パラメーターに値を指定しないと、デフォルトは db2gse.gse_enable_autogc ストアード・プロシージャーを呼び出したユーザー ID になります。</p>

使用すべきでないストアード・プロシージャ

表 67. db2gse.gse_enable_autogc ストアード・プロシージャの入力パラメーター (続き)

名前	データ・タイプ	説明
layerTable	VARCHAR(128)	このストアード・プロシージャにより作成または再活性化されるトリガーが、操作する表の名前。 このパラメーターは NULL にはできません。
layerColumn	VARCHAR(128)	このストアード・プロシージャが作成または再活性化するトリガーにより保守される、地理情報列の名前。 このパラメーターは NULL にはできません。 layerColumn パラメーターは、表のレイヤーとして登録された列を参照する必要があります。
gcId	INTEGER	このストアード・プロシージャが作成または再活性化する挿入トリガーおよび更新トリガーが呼び出す、ジオコーダーの ID。 operMode パラメーターが GSE_AUTOGC_CREATE になっている場合、このパラメーターは NULL にはできません。operMode が GSE_AUTOGC_RECREATE に設定されていれば、NULL にできます。
precisionLevel	INTEGER	ソース・データが対応する参照データとどの程度一致すれば、ジオコーダーがソース・データを正常に処理できるかを示す値。 operMode パラメーターが GSE_AUTOGC_CREATE になっている場合、このパラメーターは NULL にはできません。operMode が GSE_AUTOGC_RECREATE に設定されていれば、NULL にできます。 精度レベルは 1 から 100 % の範囲にできます。
vendorSpecific	VARCHAR(256)	ベンダーが提供する技術情報。たとえば、パラメーターを設定するためにベンダーが使用するファイルのパスと名前など。 operMode パラメーターが GSE_AUTOGC_CREATE になっている場合、このパラメーターは NULL にはできません。operMode が GSE_AUTOGC_RECREATE に設定されていれば、NULL にできます。

結果:

表 68. db2gse.gse_enable_autogc ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。

使用すべきでないストアード・プロシージャ

表 68. db2gse.gse_enable_autogc ストアード・プロシージャの出力パラメーター (続き)

名前	データ・タイプ	説明
msgText	VARCHAR(1024)	サーバーで作成される、完全なエラー・メッセージ。

db2gse.gse_enable_db

このストアード・プロシージャは、空間データを保管し、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ために使用します。これらのリソースには、空間データ、地理情報索引タイプ、カタログ表およびビュー、提供された関数、およびその他のストアード・プロシージャが含まれます。このストアード・プロシージャの外部ライブラリーおよび関数名は、db2gse.gse_enable_db です。

許可:

このストアード・プロシージャ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 SYSADM または DBADM 権限のいずれかを持つ必要があります。

結果:

表 69. db2gse.gse_enable_db ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

db2gse.gse_enable_idx

このストアード・プロシージャは、地理情報列の索引を作成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。

- 使用可能索引を使用する表が入っているデータベースに対する、SYSADM 権限または DBADM 権限。
- この表に対する CONTROL または INDEX 特権。

パラメーター:

表 70. db2gse.gse_enable_idx ストアード・プロシージャの入力パラメーター

名前	データ・タイプ	説明
layerSchema	VARCHAR(30)	layerTable パラメーターに指定された表が属するスキーマの名前。 このパラメーターは NULL にできます。 このパラメーターには値を指定する必要があります。パラメーターは NULL 値にすることができません。
layerTable	VARCHAR(128)	作成する索引を定義する表の名前。 このパラメーターは NULL にはできません。
layerColumn	VARCHAR(128)	作成する索引を使用して検索する、地理情報を使用できる列の名前。 このパラメーターは NULL にはできません。
indexName	VARCHAR(128)	作成する索引の名前。 このパラメーターは NULL にはできません。 スキーマ名は指定しないでください。DB2 Spatial Extender は、layerSchema パラメーターで指定されたスキーマに自動的に索引を割り当てます。
gridSize1	DOUBLE	最も細かい索引グリッドの細分度をどの程度にするかを示す数値。 このパラメーターは NULL にはできません。
gridSize2	DOUBLE	次のいずれかを示す値。(1) この索引には 2 番目のグリッドはない、または (2) 2 番目のグリッドの細分度をどのようにするか。 このパラメーターは NULL にできます。 2 番目のグリッドを持たない場合は 0 を指定します。2 番目のグリッドが必要な場合は、gridSize1 で示されたグリッドよりも低い細分度にする必要があります。
gridSize3	DOUBLE	次のいずれかを示す値。(1) この索引には 3 番目のグリッドはないことを示す数、または (2) 3 番目のグリッドの細分度を何にするかを示す数。 このパラメーターは NULL にできます。 3 番目のグリッドを持たない場合は 0 を指定します。3 番目のグリッドが必要な場合は、gridSize2 で示されたグリッドよりも低い細分度にする必要があります。

使用すべきでないストアード・プロシージャ

結果:

表 71. *db2gse.gse_enable_idx* ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

db2gse.gse_enable_sref

このストアード・プロシージャは、指定された座標系の負の数および小数を、DB2 Spatial Extender が保管できるように、どのように正の整数に変換するかを指定します。ユーザーの指定をまとめて、**空間参照系** と呼びます。このストアード・プロシージャを処理すると、空間参照系についての情報が DB2GSE.SPATIAL_REF_SYS カタログ・ビューに追加されます。

許可:

必要ありません。

パラメーター:

表 72. *db2gse.gse_enable_sref* ストアード・プロシージャの入力パラメーター

名前	データ・タイプ	説明
srId	INTEGER	空間参照系の数値 ID。 この ID は、地理情報を使用可能にしたデータベース内でユニークである必要があります。 このパラメーターは NULL にはできません。
srName	VARCHAR(64)	空間参照系の簡略説明。 この説明は、地理情報を使用可能にしたデータベース内でユニークである必要があります。 このパラメーターは NULL にはできません。
falseX	DOUBLE	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。 このパラメーターは NULL にはできません。
falseY	DOUBLE	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。 このパラメーターは NULL にはできません。
xyunits	DOUBLE	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。 このパラメーターは NULL にはできません。

表 72. db2gse.gse_enable_sref ストアード・プロシージャの入力パラメーター (続き)

名前	データ・タイプ	説明
falsez	DOUBLE	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。 このパラメーターは NULL にはできません。
zunits	DOUBLE	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。 このパラメーターは NULL にはできません。
falsem	DOUBLE	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。 このパラメーターは NULL にはできません。
munits	DOUBLE	小数の指標と乗算すると、32 ビットのデータ項目として保管できる整数になる数。 このパラメーターは NULL にはできません。
scId	INTEGER	空間参照系の導出元の座標系の数値 ID。座標系の数値 ID を見つけるには、DB2GSE.COORD_REF_SYS カタログ・ビューを参照してください。 このパラメーターは NULL にはできません。

結果:

表 73. db2gse.gse_enable_sref ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

db2gse.gse_export_shape

このストアード・プロシージャは、レイヤーおよびこれに関連する表を形状ファイルにエクスポートするため、または、新しい形状ファイルを作成し、レイヤーおよびそれに関連する表をこの新しい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・プロシージャを呼び出すユーザー ID は、エクスポートされる表に対する SELECT 特権を持つ必要があります。

使用すべきでないストアード・プロシージャ

パラメーター:

表 74. *db2gse.gse_export_shape* ストアード・プロシージャの入力パラメーター

名前	データ・タイプ	説明
layerSchema	VARCHAR(30)	layerTable パラメーターに指定された表が属するスキーマの名前。 このパラメーターは NULL にできます。 layerSchema パラメーターに値を指定しないと、デフォルトで、 <i>db2gse.gse_export_shape</i> ストアード・プロシージャを呼び出したユーザー ID になります。
layerTable	VARCHAR(128)	エクスポートする表の名前。 このパラメーターは NULL にはできません。
layerColumn	VARCHAR(30)	エクスポートするレイヤーとして登録された列の名前。 このパラメーターは NULL にはできません。
fileName	VARCHAR(128)	指定されたレイヤーのエクスポート先である形状ファイルの名前。 このパラメーターは NULL にはできません。
whereClause	VARCHAR(1024)	whereClause の本体。これは、エクスポートされる行のセットに関する制限を定義します。この文節は、エクスポートする表内のどの属性列でも参照することができます。この文節にはキーワード WHERE は必要ありません。 このパラメーターは NULL にできます。

結果:

表 75. *db2gse.gse_export_shape* ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

制限:

1 度にエクスポートできるレイヤーは 1 つだけです。

db2gse.gse_disable_autogc

このストアード・プロシージャは、地理情報列をそれに関連する属性列 (複数可) と同期化させるためのトリガーをドロップ、または一時的に使用不可にする場合に使用します。たとえば、属性列の値をバッチ・モードでジオコーディングしている間は、トリガーを使用不可にすることをお勧めします。

許可:

このストアード・プロシージャを呼び出すユーザー ID は、次の権限、特権、または特権のセットのいずれかを持つ必要があります。

- ドロップまたは一時的に使用不可にされるトリガーが定義されている表が入っているデータベースに関する、SYSADM 権限または DBADM 権限。
- この表に対する CONTROL 特権。
- この表に対する ALTER 特権および UPDATE 特権。

注: CONTROL および ALTER 特権の場合、DB2GSE スキーマに対する DROPIN 権限を持つ必要があります。

パラメーター:

表 76. db2gse.gse_disable_autogc ストアード・プロシージャの入力パラメーター

名前	データ・タイプ	説明
operMode	SMALLINT	<p>トリガーをドロップするのか、一時的に使用不可にするのかを示します。</p> <p>ドロップされたトリガーは SQL ステートメントに影響を与えません。</p> <p>一時的に使用不可にされたトリガーは、以前に設定したパラメーターを再指定しなくても、再作成することができます。</p> <p>このパラメーターは NULL にはできません。</p> <p>トリガーをドロップするには、GSE_AUTOGC_DROP マクロを使用します。トリガーを一時的に使用不可にするには、GSE_AUTOGC_INVALIDATE マクロを使用します。これらのマクロに関連する値については、db2gse.h ファイルを参照してください。AIX では、このファイルは \$DB2INSTANCE/sqlib/include/ ディレクトリーにあります。Windows NT では、これは %DB2PATH%\include\ ディレクトリーにあります。</p>
layerSchema	VARCHAR(30)	<p>layerTable 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。</p> <p>このパラメーターは NULL にできます。</p> <p>layerSchema パラメーターに値を指定しないと、デフォルトで db2gse.gse_disable_autogc ストアード・プロシージャを呼び出したユーザー ID になります。</p>
layerTable	VARCHAR(128)	<p>ドロップする、または一時的に使用不可にするトリガーが定義された表の名前。</p> <p>このパラメーターは NULL にはできません。</p>

使用すべきでないストアード・プロシージャ

表 76. *db2gse.gse_disable_autogc* ストアード・プロシージャの入力パラメーター (続き)

名前	データ・タイプ	説明
layerColumn	VARCHAR(128)	ドロップまたは一時的に使用不可にするトリガーにより保守されている、地理情報使用可能列の名前。 このパラメーターは NULL にはできません。

結果:

表 77. *db2gse.gse_disable_autogc* ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

db2gse.gse_disable_db

このストアード・プロシージャは、DB2 Spatial Extender が、空間データを保管し、このデータに対して行われる操作をサポートするために必要なリソースを除去するために使用します。

このストアード・プロシージャは、データベースを地理情報操作に使用可能にした後で、地理情報の表の列やデータを追加する前に発生した問題などを解決することを目的としています。たとえば、データベースを地理情報操作に使用できるようにした後、それに代えて DB2 Spatial Extender を他のデータベースに使用することにしたとします。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・プロシージャを呼び出して、最初のデータベースからすべての地理情報リソースを除去することができます。

許可:

このストアード・プロシージャ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、SYSADM または DBADM 権限のいずれかを持つ必要があります。

結果:

表 78. *db2gse.gse_disable_db* ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

db2gse.gse_disable_sref

このストアード・プロシージャは、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・プロシージャを処理すると、空間参照系についての情報が DB2GSE.SPATIAL_REF_SYS カタログ・ビューから除去されます。

前提条件:

空間参照系をドロップするには、その前にそれを使用するすべてのレイヤーの登録を抹消しておく必要があります。このようなレイヤーの登録が残っていると、空間参照系をドロップする要求はリジェクトされます。

許可:

必要ありません。

処理:

表 79. db2gse.gse_disable_sref ストアード・プロシージャの入力パラメーター

名前	データ・タイプ	説明
srId	INTEGER	ドロップされる空間参照系の数値 ID。 このパラメーターは NULL にはできません。

結果:

表 80. db2gse.gse_disable_sref ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

db2gse.gse_import_shape

このストアード・プロシージャは、地理情報操作用に使用可能になっているデータベースに、ESRI 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・プロシージャは、次の 2 つの方法のいずれかで操作することができます。

- 形状ファイルの宛先が、登録されたレイヤーの列を持つ既存の表の場合、DB2 Spatial Extender はファイルのデータを表にロードします。
- 形状ファイルの宛先が存在しない表の場合、DB2 Spatial Extender は、地理情報列を持つ表を作成し、その列をレイヤーとして登録し、レイヤーおよび表のその他の列にファイルのデータをロードします。

ESRI 形状表記のセットをインポートすると、少なくとも 2 つのファイルを受け取ります。それらのファイルのファイル名の接頭部はすべて同じですが、拡張子が異なります。たとえば、必ず受け取る 2 つのファイルの拡張子は、.shp および .shx です。

使用すべきでないストアード・プロシージャ

形状表記のセットのファイルを受け取るには、ファイルが共通に持つ名前を `fileName` パラメーターに割り当てます。拡張子は指定しないでください。この方法により、必要なすべてのファイル (.shp ファイル、.shx ファイル、および含まれる可能性のあるその他のファイル) を確実にインポートすることができます。

たとえば、ESRI 形状表記のセットが `Lakes.shp` および `Lakes.shx` と呼ばれるファイルに保管されているとします。これらの表記をインポートする場合、名前 `Lakes` のみを `fileName` パラメーターに指定します。

SDE 転送ファイルは、名前を持ちますが拡張子はありません。したがって SDE 転送ファイルをインポートする場合は、この名前を拡張子なしで `fileName` パラメーターに指定します。

許可:

このストアード・プロシージャ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。

- インポートされた形状データがロードされる表が入っているデータベースに対する SYSADM 権限または DBADM 権限。
- この表に対する CONTROL 特権。

パラメーター:

表 81. `db2gse.gse_import_shape` ストアード・プロシージャの入力パラメーター

名前	データ・タイプ	説明
<code>layerSchema</code>	VARCHAR(30)	<code>layerTable</code> 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。 このパラメーターは NULL にできます。 <code>layerSchema</code> パラメーターの値を指定しないと、デフォルトで <code>db2gse.gse_import_shape</code> ストアード・プロシージャを呼び出したユーザー ID になります。
<code>layerTable</code>	VARCHAR(128)	インポートされる形状ファイルをロードする表の名前。 このパラメーターは NULL にはできません。
<code>layerColumn</code>	VARCHAR(30)	形状データのロード先の、レイヤーとして登録されている列の名前。 このパラメーターは NULL にはできません。
<code>fileName</code>	VARCHAR(128)	インポートされる形状ファイルの名前。 このパラメーターは NULL にはできません。

表 81. db2gse.gse_import_shape ストアード・プロシージャの入力パラメーター (続き)

名前	データ・タイプ	説明
exceptionFile	VARCHAR(128)	インポートできなかった形状が保管されるファイルのパスおよび名前。これは、db2gse.gse_import_shape ストアード・プロシージャの実行時に作成される新しいファイルです。 ファイル名を、拡張子を付けずに、exceptionFile パラメーターに指定します。 このパラメーターは NULL にはできません。
srId	INTEGER	形状データのロード先のレイヤーに使用される空間参照系の ID。 このパラメーターは NULL にできます。 この ID を指定しないと、内部トランスフォーメーションは、形状ファイルに可能な最大解像度に設定されます。
commitScope	INTEGER	チェックポイントあたりのレコード数。 このパラメーターは NULL にできます。

結果:

表 82. db2gse.gse_import_shape ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

db2gse.gse_register_gc

このストアード・プロシージャは、デフォルト以外のジオコーダーの登録に使用します。ジオコーダーがすでに登録されているかどうかを調べるには、DB2GSE.SPATIAL_GEOCODER 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・プロシージャを呼び出すユーザー ID は、このストアード・プロシージャが登録するジオコーダーを含むデータベースに対して、SYSADM または DBADM 権限のいずれかを持つ必要があります。

使用すべきでないストアード・プロシージャ

パラメーター:

表 83. db2gse.gse_register_gc ストアード・プロシージャの入力パラメーター

名前	データ・タイプ	説明
gcId	INTEGER	登録するジオコーダーの数値 ID。 この ID は、データベース内でユニークである必要があります。 このパラメーターは NULL にはできません。
gcName	VARCHAR(64)	登録するジオコーダーの簡略説明。 この説明は、データベース内でユニークな文字ストリングである必要があります。 このパラメーターは NULL にはできません。
vendorName	VARCHAR(64)	登録するジオコーダーの提供者であるベンダーの名前。 このパラメーターは NULL にはできません。
primaryUDF	VARCHAR(256)	登録するジオコーダーの完全修飾名。 このパラメーターは NULL にはできません。
precisionLevel	INTEGER	ソース・データが対応する参照データとどの程度一致すれば、ジオコーダーがソース・データを正常に処理できるかを示す値。 精度レベルは 1 から 100 % の範囲にできます。 このパラメーターは NULL にはできません。
vendorSpecific	VARCHAR(256)	ベンダーが提供する技術情報。たとえば、パラメーターを設定するためにベンダーが使用するファイルのパスと名前など。 このパラメーターは NULL にできます。
geoArea	VARCHAR(256)	ジオコーディングされる地理上のエリア。 このパラメーターは NULL にできます。
説明	VARCHAR(256)	ベンダーが提供する注釈 このパラメーターは NULL にできます。

結果:

表 84. db2gse.gse_register_gc ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

db2gse.gse_register_layer

このストアード・プロシージャは、地理情報列をレイヤーとして登録するために使用します。このストアード・プロシージャが処理されると、登録されるレイヤーの情報が DB2GSE.GEOMETRY_COLUMNS 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。

- 表レイヤーの場合:
 - レイヤーが属する表が入っているデータベースに対する SYSADM 権限または DBADM 権限。
 - この表に対する CONTROL または ALTER 特権。
- ビューのレイヤーの場合:
 - 以下を含む基本表または表に対する SELECT 特権 (1) このレイヤー用にジオコーディングされるアドレス・データ (2) ジオコーディングの結果の空間データ

パラメーター:

表 85. db2gse.gse_register_layer ストアード・プロシージャの入力パラメーター

名前	データ・タイプ	説明
layerSchema	INTEGER(30)	layerTable 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。 このパラメーターは NULL にできます。 layerSchema パラメーターに値を指定しないと、デフォルトで db2gse.gse_register_layer ストアード・プロシージャを呼び出したユーザー ID になります。
layerTable	VARCHAR(128)	レイヤーとして登録される列が入っている表またはビューの名前。 このパラメーターは NULL にはできません。
layerColumn	VARCHAR(128)	レイヤーとして登録される列の名前。表の場合、この列が存在しないと、DB2 Spatial Extender は ALTER ステートメントを使用してこれを追加します。ビューの場合、この列がすでに存在していなければなりません。 layerColumn パラメーターには 1 つの列だけを指定することができます。したがって、表またはビューの複数の列をレイヤーとして登録する場合は、それぞれの列ごとに、このストアード・プロシージャを実行する必要があります。 このパラメーターは NULL にはできません。

使用すべきでないストアード・プロシージャ

表 85. db2gse.gse_register_layer ストアード・プロシージャの入力パラメーター (続き)

名前	データ・タイプ	説明
layerTypeName	VARCHAR(64)	<p>レイヤーとして登録される列のデータ・タイプ。DB2 Spatial Extender により提供されたデータ・タイプのみが受け付けられます。データ・タイプは次の例のように、大文字で指定する必要があります。</p> <p>ST_POINT</p> <p>スキーマ名は自動的に追加されるので、指定する必要はありません。</p> <p>この列が、このストアード・プロシージャの処理時に作成される表の列である場合、このパラメーターは NULL にはできません。それ以外の場合、列が表またはビューの既存の列である場合は、このパラメーターは NULL にできます。</p>
srId	INTEGER	<p>このレイヤーに使用される空間参照系の ID。</p> <p>表のレイヤーの場合、このパラメーターは NULL にはできません。ビューのレイヤーを登録する場合は、DB2 Spatial Extender はこのパラメーターを無視します。</p>
geoSchema	VARCHAR(30)	<p>この列が属するビューの基礎にある表のスキーマ。このパラメーターは、ビューの列をレイヤーとして登録する場合に適用されます。</p> <p>このパラメーターは、ビューの列をレイヤーとして登録する場合は NULL にできます。表の列をレイヤーとして登録する場合は、DB2 Spatial Extender はこのパラメーターを無視します。</p> <p>複数の基本表に基づくビューまたは、他のビューに基づくビューは、このパラメーターではサポートされません。</p> <p>geoSchema パラメーターの値を指定しないと、デフォルトは layerSchema パラメーターの値になります。</p>
geoTable	VARCHAR(128)	<p>この列が属するビューの基礎にある表の名前。このパラメーターは、ビューの列をレイヤーとして登録する場合に適用されます。</p> <p>複数の基本表に基づくビューまたは、他のビューに基づくビューは、このパラメーターではサポートされません。</p> <p>このパラメーターは、ビューの列をレイヤーとして登録する場合は NULL にできません。表の列をレイヤーとして登録する場合は、DB2 Spatial Extender はこのパラメーターを無視します。</p>

使用すべきでないストアド・プロシージャ

表 85. db2gse.gse_register_layer ストアド・プロシージャの入力パラメーター (続き)

名前	データ・タイプ	説明
geoColumn	VARCHAR(128)	<p>このビューの列の基礎にある表の列の名前。このパラメーターは、ビューの列をレイヤーとして登録する場合に適用されます。</p> <p>複数の基本表に基づくビューまたは、他のビューに基づくビューは、このパラメーターではサポートされません。</p> <p>このパラメーターは、ビューの列をレイヤーとして登録する場合は NULL にできません。表の列をレイヤーとして登録する場合は、DB2 Spatial Extenderはこのパラメーターを無視します。</p>
nAttributes	SMALLINT	<p>このレイヤー用にジオコーディングされるソース・データを含む列の数。</p> <p>このパラメーターは、表の列をレイヤーとして登録する場合は NULL にできます。ビューの列をレイヤーとして登録する場合は、DB2 Spatial Extenderはこのパラメーターを無視します。</p>
attr1Name	VARCHAR(128)	<p>このレイヤーにジオコーディングされるソース・データが入っている最初の列の名前。</p> <p>このパラメーターは、表の列をレイヤーとして登録する場合は NULL にできます。ビューの列をレイヤーとして登録する場合は、DB2 Spatial Extenderはこのパラメーターを無視します。</p> <p>デフォルトのジオコーダーを使用する場合は、attr1Name 列に番地を保管する必要があります。</p>
attr2Name	VARCHAR(128)	<p>このレイヤーにジオコーディングされるソース・データが入っている 2 番目の列の名前。</p> <p>このパラメーターは、表の列をレイヤーとして登録する場合は NULL にできます。ビューの列をレイヤーとして登録する場合は、DB2 Spatial Extenderはこのパラメーターを無視します。</p> <p>デフォルトのジオコーダーを使用する場合は、attr2Name 列に市の名前を保管する必要があります。</p>
attr3Name	VARCHAR(128)	<p>このレイヤーにジオコーディングされるソース・データが入っている 3 番目の列の名前。</p> <p>このパラメーターは、表の列をレイヤーとして登録する場合は NULL にできます。ビューの列をレイヤーとして登録する場合は、DB2 Spatial Extenderはこのパラメーターを無視します。</p> <p>デフォルトのジオコーダーを使用する場合は、attr3Name 列に州の名前または省略形を保管する必要があります。</p>

使用すべきでないストアード・プロシージャ

表 85. db2gse.gse_register_layer ストアード・プロシージャの入力パラメーター (続き)

名前	データ・タイプ	説明
attr4Name	VARCHAR(128)	<p>このレイヤーにジオコーディングされるソース・データが入っている 4 番目の列の名前。</p> <p>このパラメーターは、表の列をレイヤーとして登録する場合は NULL にできます。ビューの列をレイヤーとして登録する場合は、DB2 Spatial Extender はこのパラメーターを無視します。</p> <p>デフォルトのジオコーダーを使用する場合は、attr4Name 列に郵便番号を保管する必要があります。</p>
attr5Name	VARCHAR(128)	<p>このレイヤーにジオコーディングされるソース・データが入っている 5 番目の列の名前。</p> <p>このパラメーターは、表の列をレイヤーとして登録する場合は NULL にできます。ビューの列をレイヤーとして登録する場合は、DB2 Spatial Extender はこのパラメーターを無視します。</p> <p>デフォルトのジオコーダーは Attr5Name 列を無視します。</p>
attr6Name	VARCHAR(128)	<p>このレイヤーにジオコーディングされるソース・データが入っている 6 番目の列の名前。</p> <p>このパラメーターは、表の列をレイヤーとして登録する場合は NULL にできます。ビューの列をレイヤーとして登録する場合は、DB2 Spatial Extender はこのパラメーターを無視します。</p> <p>デフォルトのジオコーダーは Attr6Name 列を無視します。</p>
attr7Name	VARCHAR(128)	<p>このレイヤーにジオコーディングされるソース・データが入っている 7 番目の列の名前。</p> <p>このパラメーターは、表の列をレイヤーとして登録する場合は NULL にできます。ビューの列をレイヤーとして登録する場合は、DB2 Spatial Extender はこのパラメーターを無視します。</p> <p>デフォルトのジオコーダーは Attr7Name 列を無視します。</p>
attr8Name	VARCHAR(128)	<p>このレイヤーにジオコーディングされるソース・データが入っている 8 番目の列の名前。</p> <p>このパラメーターは、表の列をレイヤーとして登録する場合は NULL にできます。ビューの列をレイヤーとして登録する場合は、DB2 Spatial Extender はこのパラメーターを無視します。</p> <p>デフォルトのジオコーダーは Attr8Name 列を無視します。</p>

表 85. db2gse.gse_register_layer ストアード・プロシージャの入力パラメーター (続き)

名前	データ・タイプ	説明
attr9Name	VARCHAR(128)	このレイヤーにジオコーディングされるソース・データが入っている 9 番目の列の名前。 このパラメーターは、表の列をレイヤーとして登録する場合は NULL にできます。ビューの列をレイヤーとして登録する場合は、DB2 Spatial Extender はこのパラメーターを無視します。 デフォルトのジオコーダーは Attr9Name 列を無視します。
attr10Name	VARCHAR(128)	このレイヤーにジオコーディングされるソース・データが入っている 10 番目の列の名前。 このパラメーターは、表の列をレイヤーとして登録する場合は NULL にできます。ビューの列をレイヤーとして登録する場合は、DB2 Spatial Extender はこのパラメーターを無視します。 デフォルトのジオコーダーは Attr10Name 列を無視します。

結果:

表 86. db2gse.gse_register_layer ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

制約事項:

このストアード・プロシージャは、次のタイプの表には働きません。

- A = Alias (別名)
- H = Hierarchy table (階層表)
- N = Nickname (ニックネーム)
- S = Summary table (サマリー表)
- U = Typed table (型付き表)
- W = Typed view (型付きビュー)

また、以下の制約事項があります。

- ビューの列をレイヤーとして登録する場合、すでにレイヤーとして登録された表の列に基づくものでなければなりません。
- 登録するレイヤーにジオコーディングされるデータを入れることができる属性列は、10 個までです。

db2gse.gse_run_gc

このストアード・プロシージャは、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、SYSADM 権限または DBADM 権限。
- この表に関する CONTROL 特権または UPDATE 特権

パラメーター:

表 87. db2gse.gse_run_gc ストアード・プロシージャの入力パラメーター

名前	データ・タイプ	説明
layerSchema	VARCHAR(30)	layerTable 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。 このパラメーターは NULL にできます。 layerSchema パラメーターの値を指定しないと、デフォルトで db2gse.gse_run_gc を呼び出したユーザー ID になります。
layerTable	VARCHAR(128)	ジオコーディングされたデータを挿入する列が入っている表の名前。 このパラメーターは NULL にはできません。
layerColumn	VARCHAR(128)	ジオコーディングされたデータを挿入する列の名前。 このパラメーターは NULL にはできません。
gcId	INTEGER	実行するジオコーダーの ID。 このパラメーターは NULL にできます。 登録されたジオコーダーの ID を調べるには、DB2GSE.SPATIAL_GEOCODER カタログ・ビューを参照してください。
precisionLevel	INTEGER	ソース・データが対応する参照データとどの程度一致すれば、ジオコーダーがソース・データを正常に処理できるかを示す値。 このパラメーターは NULL にできます。 精度レベルは 1 から 100 % の範囲にできます。
vendorSpecific	VARCHAR(256)	ベンダーが提供する技術情報。たとえば、パラメーターを設定するためにベンダーが使用するファイルのパスと名前など。 このパラメーターは NULL にできます。

表 87. db2gse.gse_run_gc ストアード・プロシージャの入力パラメーター (続き)

名前	データ・タイプ	説明
whereClause	VARCHAR(256)	WHERE 文節の本体。これは、ジオコーディングされるレコードのセットに対する制限を定義します。この文節は、ジオコーダーの操作対象の表にある、任意の属性列を参照することができます。 このパラメーターは NULL にできます。
commitScope	INTEGER	チェックポイントあたりのレコード数。 このパラメーターは NULL にできます。

結果:

表 88. db2gse.gse_run_gc ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

db2gse.gse_unregist_gc

このストアード・プロシージャは、デフォルト以外のジオコーダーの登録の抹消に使用します。登録を抹消するジオコーダーについての情報を見つけるには、DB2GSE.SPATIAL_GEOCODER 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、SYSADM または DBADM 権限のいずれかを持つ必要があります。

パラメーター:

表 89. db2gse.gse_unregist_gc ストアード・プロシージャの入力パラメーター

名前	データ・タイプ	説明
gcId	INTEGER	登録を抹消するジオコーダーの ID。 このパラメーターは NULL にはできません。

結果:

表 90. db2gse.gse_unregist_gc ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

db2gse.gse_unregist_layer

このストアード・プロシージャは、レイヤーの登録を抹消するために使用します。このストアード・プロシージャは、次のようにして登録を抹消します。

- DB2 Spatial Extender カタログ表からレイヤーの定義を除去する。
- レイヤーの空間データが、レイヤーの空間参照系の要件を必ず満たすようにするために、DB2 Spatial Extender がこのレイヤーの基本表に課したチェック制約を削除する。
- アドレス・データが追加、変更、または除去されるたびに地理情報列を更新するために使用される、トリガーをドロップする。

表の行にあるアドレス・データをジオコーディングすると、結果の空間データは同じ行に置かれます。したがって、行を削除すると、アドレス・データと空間データは同時に削除されます。トリガーは空間データを削除しません。このストアード・プロシージャが処理されると、レイヤーの情報は DB2GSE.GEOMETRY_COLUMNS カタログ・ビューから除去されます。

許可:

このストアード・プロシージャ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。

- 表レイヤーの場合:
 - このレイヤーの基本表が入っているデータベースに対する SYSADM 権限または DBADM 権限。
 - この表に対する CONTROL または ALTER 特権。
- ビューのレイヤーの場合:
 - 以下のものが入っている基本表または表に対する SELECT 特権 (1) このレイヤーにジオコーディングされるアドレス・データ、および (2) ジオコーディングの結果の空間データ

パラメーター:

表 91. db2gse.gse_unregist_layer ストアード・プロシージャの入力パラメーター

名前	データ・タイプ	説明
layerSchema	VARCHAR(30)	layerTable パラメーターに指定された表が属するスキーマの名前。 このパラメーターは NULL にできます。 layerSchema パラメーターの値を指定しないと、デフォルトで db2gse.gse_unregister_layer ストアード・プロシージャを呼び出したユーザー ID になります。 パラメーターに指定するスキーマ名、表名、ビュー名、列名、またはレイヤー名は、大文字で指定する必要があります。

使用すべきでないストアード・プロシージャ

表 91. *db2gse.gse_unregist_layer* ストアード・プロシージャの入力パラメーター (続き)

名前	データ・タイプ	説明
layerTable	VARCHAR(128)	layerColumn パラメーターで指定された列が入っている表の名前。 このパラメーターは NULL にはできません。
layerColumn	VARCHAR(128)	登録を抹消したい、レイヤーとして定義された地理情報列の名前。 このパラメーターは NULL にはできません。 layerColumn パラメーターに指定できるレイヤーは 1 つだけです。したがって、表またはビューの複数のレイヤーを登録抹消する場合は、それぞれのレイヤーごとに、このストアード・プロシージャを実行する必要があります。

結果:

表 92. *db2gse.gse_unregist_layer* ストアード・プロシージャの出力パラメーター

名前	データ・タイプ	説明
msgCode	INTEGER	このストアード・プロシージャの呼び出し元が戻すメッセージに関連付けられたコード。
msgText	VARCHAR(1024)	DB2 Spatial Extender サーバーで作成される、完全なエラー・メッセージ。

制限:

ビューのレイヤーとして定義されたビューの列が、表のレイヤーとして定義された表の列に基づく場合、ビューのレイヤーの登録を抹消してからでないと、この表のレイヤーの登録を抹消することはできません。

付録 B. 使用すべきでないカタログ・ビュー

このトピックでは、使用すべきでないカタログ・ビューの概要を説明しています。

注: 推奨事項: 新しいアプリケーションはすべて、DB2 Spatial Extender バージョン 8 で定義されたビューを使用して開発してください。現行アプリケーションについても、バージョン 8 で定義されたビューを使用するように更新してください。バージョン 7 で定義された、文書化されていないカタログ表を参照するアプリケーションは、バージョン 8 にマイグレーションすると働かないので、バージョン 8 の文書化されたカタログ・ビューを使用するように変更してください。

DB2GSE.COORD_REF_SYS

データベースを地理情報操作に使用できるようにすると、DB2 Spatial Extender は、使用できる座標系をカタログ表に登録します。この表から選択された列で、DB2GSE.COORD_REF_SYS カタログ・ビューが構成され、それが以下の表で説明されています。

表 93. DB2GSE.COORD_REF_SYS カタログ・ビューの列

名前	データ・タイプ	NULL 可能か ?	説明
CSID	INTEGER	はい	この座標系のユニークな数値 ID。V8 の管理インターフェースを使用して座標系を作成した場合は、CSID は記録されず、代わりに NULL が使用されます。
CS_NAME	VARCHAR(64)	いいえ	この座標系の名前。
AUTH_NAME	VARCHAR(256)	はい	この座標系をまとめた組織が準拠する組織の名前。たとえば、European Petroleum Survey Group (EPSG)。
AUTH_SRID	INTEGER	はい	AUTH_NAME 列に指定された組織が、この座標系に割り当てた数値 ID。
DESC	VARCHAR(256)	はい	この座標系の説明。
SRTEXT	VARCHAR(2048)	いいえ	この座標系の注釈テキスト。

DB2GSE.GEOMETRY_COLUMNS

レイヤーを作成すると、DB2 Spatial Extender は、レイヤーの ID およびレイヤーに関する情報を、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.GEOMETRY_COLUMNS カタログ・ビューが構成され、それが以下の表で説明されています。

使用すべきでないカタログ・ビュー

表 94. DB2GSE.GEOMETRY_COLUMNS カタログ・ビューの列

名前	データ・タイプ	NULL 可能か ?	説明
LAYER_CATALOG	VARCHAR(30)	はい	NULL DB2 Spatial Extender には、LAYER_CATALOG の概念はありません。
LAYER_SCHEMA	VARCHAR(30)	いいえ	このレイヤーとして登録された列が入っている、表またはビューのスキーマ。
LAYER_TABLE	VARCHAR(128)	いいえ	このレイヤーとして登録された列を含む、表またはビューの名前。
LAYER_COLUMN	VARCHAR(128)	いいえ	このレイヤーとして登録された列の名前。
GEOMETRY_TYPE	INTEGER	はい	このレイヤーとして登録された列のデータ・タイプ。列が、Spatial Extender により定義されたいずれかの図形タイプのユーザー定義サブタイプを持つ場合は、この値は NULL です。
SRID	INTEGER	いいえ	このレイヤーとして登録された列の値に使用される、空間参照系の ID。
STORAGE_TYPE	INTEGER	はい	NULL

DB2GSE.SPATIAL_GEOCODER

使用できるジオコーダーは、カタログ表に登録されます。この表から選択された列で DB2GSE.SPATIAL_GEOCODER 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 95. DB2GSE.SPATIAL_GEOCODER カタログ・ビューの列

名前	データ・タイプ	NULL 可能か ?	説明
GCID	INTEGER	いいえ	ジオコーダーの数値 ID。
GC_NAME	VARCHAR(64)	いいえ	ジオコーダーの名前 ID。
VENDOR_NAME	VARCHAR(128)	いいえ	ジオコーダーを提供したベンダーの名前。
PRIMARY_UDF	VARCHAR(256)	いいえ	ジオコーダーの完全修飾名。
PRECISION_LEVEL	INTEGER	いいえ	ソース・データが対応する参照データとどの程度一致すれば、ジオコーダーがソース・データを正常に処理できるかを示す値。
VENDOR_SPECIFIC	VARCHAR(256)	はい	ジオコーダーがサポートする特別なパラメーターを設定するために、ベンダーが使用できるファイルへのパスおよびファイル名。
GEO_AREA	VARCHAR(256)	はい	ジオコーディングされるロケーションが入っている地理上のエリア。
説明	VARCHAR(256)	はい	ジオコーダーの説明。

DB2GSE.SPATIAL_REF_SYS

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atalog・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 96. DB2GSE.SPATIAL_REF_SYS カatalog・ビューの列

名前	データ・タイプ	NULL 可能か ?	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	この空間参照系の注釈テキスト。
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。
MUNITS	FLOAT	いいえ	小数の指標と乗算すると、32 ビットのデータ項目として保管できる整数になる数。

使用すべきでないカタログ・ビュー

付録 C. 使用すべきでない空間処理関数

このトピックには、使用すべきでない関数がまとめられています。以下の表には、使用すべきでないすべての空間処理関数および、これに代わるバージョン 8 の新しい関数がリストされています。

表 97. 使用すべきでない関数および代わりとなる新しい関数

使用すべきでない関数	新しい関数
AsShape	ST_AsShape
GeometryFromShape	ST_Geometry
Is3D	ST_Is3D
IsMeasured	ST_IsMeasured
LineFromShape	ST_LineString
LocateAlong	ST_FindMeasure
LocateBetween	ST_MeasureBetween
M	ST_M
MLine FromShape	ST_MultiLineString
MPointFromShape	ST_MultiPoint
MPolyFromShape	ST_MultiPolygon
PointFromShape	ST_Point
PolyFromShape	ST_Polygon
ShapeToSQL	ST_Geometry
ST_GeomFromText	ST_Geometry
ST_GeomFromWKB	ST_Geometry
ST_LineFromText	ST_LineString
ST_LineFromWKB	ST_LineString
ST_MLineFromText	ST_MultiLineString
ST_MLineFromWKB	ST_MultiLineString
ST_MPointFromText	ST_MultiPoint
ST_MPointFromWKB	ST_MultiPoint
ST_MPolyFromText	ST_MultiPolygon
ST_MPolyFromWKB	ST_MultiPolygon
ST_OrderingEquals	
ST_Point(Double、 Double、 db2gse.coordref)	ST_Point(Double、 Double、 Integer)
ST_PointFromText	ST_Point
ST_PolyFromText	ST_Polygon
ST_PolyFromWKB	ST_Polygon
ST_Transform(Double、 Double、 db2gse.coordref)	ST_Transform(ST_Geometry、 Integer)
ST_SymmetricDiff	ST_SymDifference
Z	ST_Z

AsShape

目的:

AsShape は図形オブジェクトを入力とし、BLOB を戻します。

フォーマット:

db2gse.AsShape(g db2gse.ST_Geometry)

結果:

BLOB(1m)

GeometryFromShape

目的:

GeometryFromShape は、形状および空間参照系 ID を入力とし、図形オブジェクトを戻します。

フォーマット:

db2gse.GeometryFromShape(ShapeGeometry Blob(1M)、SRID db2gse.coordref)

結果:

db2gse.ST_Geometry

Is3d

目的:

Is3d は図形オブジェクトを入力とし、オブジェクトが 3D 座標を持つ場合は 1 を返し、それ以外の場合は 0 を戻します。

フォーマット:

db2gse.Is3d(g db2gse.ST_Geometry)

結果:

Integer

IsMeasured

目的:

IsMeasured は図形オブジェクトを入力とし、そのオブジェクトが指標を持つ場合は 1 を返し、それ以外は 0 を返します。

フォーマット:

```
db2gse.IsMeasured(g db2gse.ST_Geometry)
```

結果:

Integer

LineFromShape

目的:

LineFromShape は、タイプがポイントの形状および空間参照系 ID を入力とし、折れ線を返します。

フォーマット:

```
db2gse.Line FromShape(ShapeLineString Blob(1M), SRID db2gse.coordref)
```

結果:

db2gse.ST_LineString

LocateAlong

目的:

LocateAlong は、図形オブジェクトおよび指標を入力とし、指標にあるポイントのセットを複数ポイントとして返します。

複数ポイントおよび指標が入力として LocateAlong に与えられ、複数ポイントにその指標が含まれていない場合、LocateAlong は POINT EMPTY を返します。

フォーマット:

```
db2gse.LocateAlong(g db2gse.ST_Geometry, measure Double)
```

結果:

db2gse.ST_Geometry

LocateBetween

目的:

LocateBetween は、図形オブジェクトと 2 つの指標ロケーションを入力とし、この 2 つの指標ロケーションの間の、つながっていないパスのセットを現す図形を返します。

使用すべきでない空間処理関数

フォーマット:

```
db2gse.LocateBetween(g db2gse.ST_Geometry、 measure Double、 measure Double)
```

結果:

```
db2gse.ST_Geometry
```

M

目的:

M はポイントを入力とし、その指標を戻します。

フォーマット:

```
db2gse.M(p db2gse.ST_Point)
```

結果:

```
Double
```

MLine FromShape

目的:

MLine FromShape は、タイプが複数折れ線の形状および空間参照系 ID を入力とし、複数折れ線を戻します。

フォーマット:

```
db2gse.MLineFromShape(ShapeMultiLineString Blob(1M)、 SRID db2gse.coordref)
```

結果:

```
db2gse.ST_MultiLineString
```

MPointFromShape

目的:

MPointFromShape は、タイプが複数ポイントの形状および空間参照系 ID を入力とし、複数ポイントを戻します。

フォーマット:

```
db2gse.MPointFromShape(ShapeMultiPoint BLOB(1M)、 SRID db2gse.coordref)
```

結果:

```
db2gse.ST_MultiPoint
```

MPolyFromShape

目的:

MPolyFromShape は、タイプが複数ポリゴンの形状および空間参照系 ID を入力とし、複数ポリゴンを戻します。

フォーマット:

```
db2gse.MPolyFromShape(ShapeMultiPolygon Blob(1m)、SRID db2gse.coordref)
```

結果:

```
db2gse.ST_MultiPolygon
```

PointFromShape

目的:

PointFromShape は、タイプがポイントの形状および空間参照系 ID を入力とし、ポイントを戻します。

フォーマット:

```
db2gse.PointFromShape(db2gse.ShapePoint blob(1M)、SRID db2gse.coordref)
```

結果:

```
db2gse.ST_Point
```

PolyFromShape

目的:

PolyFromShape は、タイプがポリゴンの形状および空間参照系 ID を入力とし、ポリゴンを戻します。

フォーマット:

```
db2gse.PolyFromShape (ShapePolygon Blob(1M)、SRID db2gse.coordref)
```

結果:

```
db2gse.ST_Polygon
```

ShapeToSQL

目的:

使用すべきでない空間処理関数

ShapeToSQL は、与えられた形状表記から db2gse.ST_Geometry 値を作成します。SRID 値 0 が自動的に使用されます。

フォーマット:

```
db2gse.ShapeToSQL(ShapeGeometry blob(1M))
```

結果:

```
db2gse.ST_Geometry
```

ST_GeomFromText

目的:

ST_GeomFromText は、事前割り当てテキスト表記および空間参照系 ID を入力とし、図形オブジェクトを戻します。

フォーマット:

```
db2gse.ST_GeomFromText(geometryTaggedText Varchar(4000), SRID db2gse.coordref)
```

結果:

```
db2gse.ST_Geometry
```

ST_GeomFromWKB

目的:

ST_GeomFromWKB は、事前割り当てバイナリー表記および空間参照系 ID を入力とし、図形オブジェクトを戻します。

フォーマット:

```
db2gse.ST_GeomFromWKB(WKBGeometry Blob(1M), SRID db2gse.coordref)
```

結果:

```
db2gse.ST_Geometry
```

ST_LineFromText

目的:

ST_LineFromText は、タイプが折れ線の事前割り当てテキスト表記および空間参照系 ID を入力とし、折れ線を戻します。

フォーマット:

```
db2gse.ST_LineFromText(lineStringTaggedText Varchar(4000), SRID db2gse.coordref)
```

結果:

db2gse.ST_LineString

ST_LineFromWKB

目的:

ST_LineFromWKB は、タイプが折れ線の事前割り当てバイナリー表記および空間参照系 ID を入力とし、折れ線を戻します。

フォーマット:

db2gse.ST_LineFromWKB(WKBLineString Blob(1M)、SRID db2gse.coordref)

結果:

db2gse.ST_LineString

ST_MLineFromText

目的:

ST_MLineFromText は、タイプが複数折れ線の事前割り当てテキスト表記および空間参照系 ID を入力とし、複数折れ線を戻します。

フォーマット:

db2gse.ST_MLineFromText(multiLineStringTaggedText String、SRID db2gse.coordref)

結果:

db2gse.ST_MultiLineString

ST_MLineFromWKB

目的:

ST_MLineFromWKB は、タイプが複数折れ線の事前割り当てバイナリー表記および空間参照系 ID を入力とし、複数折れ線を戻します。

フォーマット:

db2gse.ST_MLineFromWKB(WKBMultiLineString Blob(1M)、SRID db2gse.coordref)

結果:

db2gse.ST_MultiLineString

ST_MPointFromText

目的:

ST_MPointFromText は、タイプが複数ポイントの事前割り当てテキスト表記および空間参照系 ID を入力とし、複数ポイントを戻します。

フォーマット:

```
db2gse.ST_MPointFromText(multiPointTaggedText Varchar(4000), SRID
db2gse.coordref)
```

結果:

```
db2gse.ST_MultiPoint
```

ST_MPointFromWKB

目的:

ST_MPointFromWKB は、タイプが複数ポイントの事前割り当てバイナリー表記および空間参照系 ID を入力とし、複数ポイントを戻します。

フォーマット:

```
db2gse.ST_MPointFromWKB(WKBMultiPoint Blob(1M), SRID db2gse.coordref)
```

結果:

```
db2gse.ST_MultiPoint
```

ST_MPolyFromText

目的:

ST_MPolyFromText は、タイプが複数ポリゴンの事前割り当てテキスト表記および空間参照系 ID を入力とし、複数ポリゴンを戻します。

この関数は、同じ座標を持つ複数のポリゴンを含む複数ポリゴンを入力にはできません。

フォーマット:

```
db2gse.ST_MPolyFromText(multiPolygonTaggedText Varchar(4000), SRID
db2gse.coordref)
```

結果:

```
db2gse.ST_MultiPolygon
```

ST_MPolyFromWKB

目的:

ST_MPolyFromWKB は、タイプが複数ポリゴンの事前割り当てバイナリー表記および空間参照系 ID を入力とし、複数ポリゴンを戻します。

フォーマット:

db2gse.ST_MPolyFromWKB(WKBMultiPolygon Blob(1M), SRID db2gse.coordref)

結果:

db2gse.ST_MultiPolygon

ST_OrderingEquals

目的:

ST_OrderingEquals は 2 つの図形を比較し、図形が等しく、かつ座標が同じ順序である場合は 1 (TRUE) を戻し、それ以外は 0 (FALSE) を戻します。

フォーマット:

db2gse.ST_OrderingEquals(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

結果:

Integer

ST_Point

目的:

ST_Point は、X 座標、Y 座標、および地理情報基準を入力とし、ST_Point を戻します。

フォーマット:

db2gse.ST_Point(X Double, Y Double, SRID db2gse.coordref)

結果:

db2gse.ST_Point

ST_PointFromText

目的:

使用すべきでない空間処理関数

ST_PointFromText は、タイプがポイントの事前割り当てテキスト表記および空間参照系 ID を入力とし、ポイントを戻します。

フォーマット:

```
db2gse.ST_PointFromText(pointTaggedText Varchar(4000), SRID db2gse.coordref)
```

結果:

```
db2gse.ST_Point
```

ST_PolyFromText

目的:

ST_PolyFromText は、タイプがポリゴンの事前割り当てテキスト表記および空間参照系 ID を入力とし、ポリゴンを戻します。

フォーマット:

```
db2gse.ST_PolyFromText(polygonTaggedText Varchar(4000), SRID db2gse.coordref)
```

結果:

```
db2gse.ST_Polygon
```

ST_PolyFromWKB

目的:

ST_PolyFromWKB は、タイプがポリゴンの事前割り当てバイナリー表記および空間参照系 ID を入力とし、ポリゴンを戻します。

フォーマット:

```
db2gse.ST_PolyFromWKB(WKBPolygon Blob(1M), SRID db2gse.coordref)
```

結果:

```
db2gse.ST_Polygon
```

ST_Transform

目的:

ST_Transform は、図形が現在関連付けられている空間参照系以外の空間参照系に、図形を関連付けます。

フォーマット:

```
db2gse.ST_Transform(g db2gse.ST_Geometry, SRID db2gse.coordref)
```


結果:

db2gse.ST_Geometry

ST_SymmetricDiff

目的:

ST_SymmetricDiff は、2 つの図形オブジェクトを入力とし、ソース・オブジェクトの対称差である図形オブジェクトを戻します。

ST_SymmetricDiff 関数は、同じディメンションを持つ 2 つの交差する図形の対称差 (スペースのブール論理演算 XOR) を戻します。これらの図形が等しい場合、ST_SymmetricDiff は空の図形を戻します。図形が等しくない場合は、図形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。ST_SymmetricDiff は、交差していない部分 (1 つまたは複数) を、集合として戻します (たとえば、複数ポリゴンとして)。

異なるディメンションの図形を ST_SymmetricDiff の入力にすると、NULL が戻されます。

フォーマット:

db2gse.ST_SymmetricDiff(g1 db2gse.ST_Geometry, g2 db2gse.ST_Geometry)

結果:

db2gse.ST_Geometry

Z

目的:

Z はポイントを入力とし、その Z 座標を戻します。

フォーマット:

db2gse.Z(p db2gse.ST_Point)

結果:

Double

関連資料:

- 364 ページの『ST_AsShape』
- 444 ページの『ST_MeasureBetween、ST_LocateBetween』
- 393 ページの『ST_EnvIntersects』
- 398 ページの『ST_FindMeasure または ST_LocateAlong』
- 406 ページの『ST_Geometry』
- 419 ページの『ST_Is3d』

使用すべきでない空間処理関数

- 431 ページの『ST_LineString』
- 434 ページの『ST_M』
- 462 ページの『ST_MultiLineString』
- 464 ページの『ST_MultiPoint』
- 465 ページの『ST_MultiPolygon』
- 477 ページの『ST_Point』
- 487 ページの『ST_Polygon』
- 496 ページの『ST_SymDifference』
- 507 ページの『ST_Transform』
- 517 ページの『ST_Z』

特記事項

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。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

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を提供し、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。

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せんが、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認できません。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をお願いします。

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ます、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。

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

以下は、IBM Corporation の商標です。

ACF/VTAM	iSeries
AISPO	LAN Distance
AIX	MVS
AIXwindows	MVS/ESA
AnyNet	MVS/XA
APPN	Net.Data
AS/400	NetView
BookManager	OS/390
C Set++	OS/400
C/370	PowerPC
CICS	pSeries
Database 2	QBIC
DataHub	QMF
DataJoiner	RACF
DataPropagator	RISC System/6000
DataRefresher	RS/6000
DB2	S/370
DB2 Connect	SP
DB2 Extenders	SQL/400
DB2 OLAP Server	SQL/DS
DB2 Information Integrator	System/370
DB2 Query Patroller	System/390
DB2 Universal Database	SystemView
Distributed Relational Database Architecture	Tivoli
DRDA	VisualAge
eServer	VM/ESA
Extended Services	VSE/ESA
FFST	VTAM
First Failure Support Technology	WebExplorer
IBM	WebSphere
IMS	WIN-OS/2
IMS/ESA	z/OS
	zSeries

以下は、それぞれ各社の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。

Pentium は、Intel Corporation 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または登録商標です。

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。
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]

アプリケーション
 空間 141
 空間アプリケーション
 ヘッダー・ファイルの組み込み 141
 サンプル・プログラム 143
 Spatial Extender
 ストアード・プロシージャの呼び出し 142
アプリケーション・コントロール・ヒープ・サイズ構成パラメーター 51
アプリケーション・ヒープ・サイズ・パラメーター (APPLHEAPSZ) 51
緯度、測地
 定義 169
因数、変換
 座標 75
インスタンス
 作成 39
インストール
 インスタンスの作成 39
DB2 Spatial Extender
 検査 41
 ハードウェアおよびソフトウェア要件 26
 AIX 29
 HP-UX 32
 Linux and Linux 390 37
 Solaris オペレーティング環境 34
 Windows 28
 Spatial Extender 25
インターフェース
 空間参照系の作成 77
 DB2 Spatial Extender 17
インデックス
 測地ポロノイの作成 189
インポート
 形状データ 91
 SDE 転送データ 92
エクステント
 を使用する空間参照系の作成 77
オフセット値
 新しい空間参照系の計算 77

オフセット値 (続き)
 概要 75
オフセット値およびスケール因数の単位 75

[カ行]

回転楕円体
 座標系 535
 座標系定義での 232
 定義 168
角度単位
 座標系 535
カタログ・ビュー
 ST_COORDINATE_SYSTEMS 297
 ST_GEOCODERS 301
 ST_GEOCODER_PARAMETERS 299
 ST_GEOCODING 301
 ST_GEOCODING_PARAMETERS 303
 ST_GEOMETRY_COLUMNS 298
 ST_SIZINGS 304
 ST_SPATIAL_REFERENCE_SYSTEMS 305
 ST_UNITS_OF_MEASURE 307
関数
 空間処理
 概要 309
 データ交換フォーマットの変換 309
 関数メッセージ 156
 管理通知ログ 162
極
 困むポリゴン 209
空間アプリケーション
 ストアード・プロシージャ
 アプリケーションから呼び出し 142
 ヘッダー・ファイルの組み込み 141
空間インデックス
 測地ポロノイ 185
 タイプ 103
空間エクステント
 定義 70
空間カタログ・ビュー
 Geodetic Extender によってサポートされる 223
空間カタログ・ビュー、使用すべきでない
 COORD_REF_SYS 569
 GEOMETRY_COLUMNS 569
 SPATIAL_GEOCODER 569

空間カタログ・ビュー、使用すべきでない (続き)
 SPATIAL_REF_SYS 569
空間グリッド・インデックス
 活用 128
 空間インデックス統計を分析する 118
 グリッドのレベルとサイズ 104, 106
 格子サイズを決定する 116
 索引アドバイザー・コマンド 123
 作成 111
 測地ポロノイ・インデックスとの比較 103
 利用する SQL ステートメント 114
 利用する空間処理関数 114
 CREATE INDEX ステートメント 114
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 115
空間参照系
 作成 77, 250
 説明 70
 デフォルト 71
 DB2 Spatial Extender で提供されている 73
空間参照系の ID (SRID)
 測地向けの 167, 168
空間処理関数
 概要 309
 関連したデータ・タイプ 349
 距離情報 346
 空間インデックスを活用するための使用 128
 形状の変換 309
 考慮事項 349
 索引情報 347
 座標系間のデータ変換 347
 使用すべきでない 573
図形の比較
 概要 318
 交差 323, 330
 コンテナ・リレーションシップ 320
 図形のエンベロープ 328
 同一の図形 328
 DE-9IM パターン・マトリックス・ストリング 330
図形のプロパティ 331
 境界情報 335
 空間参照系 337
 構成情報 337
 座標および指標に関する情報 331

索引

空間処理関数 (続き)

図形のプロパティ (続き)
図形内の図形 333
データ・タイプ情報 331
ディメンションに関する情報 336
生成形状の生成
ある図形の別の図形への変換 338
概要 338
既存の指標に基づいた 344
修正フォーム 345
新規空間構成 339
複数から 1 つ 343
測地での機能の違い 218
測地ポロノイ・インデックスを利用する 185, 190
データ交換フォーマットの変換
概要 310
ジオグラフィー・マークアップ言語 (GML) 表記 317
事前割り当てテキスト表現 313
事前割り当てバイナリー表現 315
ESRI 形状表記 316
例 127
和集約 518
EnvelopesIntersect 354
MBR 集約 356
ST_AppendPoint 357
ST_Area 358
ST_AsBinary 362
ST_AsGML 363
ST_AsShape 364
ST_AsText 365
ST_Boundary 366
ST_Buffer 368
ST_Centroid 371
ST_ChangePoint 372
ST_Contains 373
ST_ConvexHull 375
ST_CoordDim 376
ST_Crosses 377
ST_Difference 379
ST_Dimension 380
ST_Disjoint 381
ST_Distance 383
ST_Edge_GC_USA 386
ST_Endpoint 390
ST_Envelope 391
ST_EnvIntersects 393
ST_EqualCoordsys 394
ST_Equals 395
ST_EqualSRS 396
ST_ExteriorRing 397
ST_FindMeasure
ST_LocateAlong 398
ST_Generalize 400
ST_GeomCollection 402

空間処理関数 (続き)

ST_GeomCollFromTxt 404
ST_GeomCollFromWKB 405
ST_Geometry 406
ST_GeometryN 408
ST_GeometryType 409
ST_GeomFromText 410
ST_GeomFromWKB 411
ST_GetIndexParms 412
ST_InteriorRingN 415
ST_Intersection 416
ST_Intersects 418
ST_Is3d 419
ST_IsClosed 420
ST_IsEmpty 422
ST_IsMeasured 423
ST_IsRing 424
ST_IsSimple 425
ST_IsValid 426
ST_Length 427
ST_LineFromText 429
ST_LineFromWKB 430
ST_LineString 431
ST_LineStringN 433
ST_LocateAlong
ST_FindMeasure 398
ST_LocateBetween
ST_MeasureBetween 444
ST_M 434
ST_MaxM 435
ST_MaxX 437
ST_MaxY 439
ST_MaxZ 440
ST_MBR 442
ST_MBRIntersects 443
ST_MeasureBetween
ST_LocateBetween 444
ST_MidPoint 446
ST_MinM 447
ST_MinX 448
ST_MinY 450
ST_MinZ 451
ST_MLineFromText 453
ST_MLineFromWKB 454
ST_MPointFromText 456
ST_MPointFromWKB 457
ST_MPolyFromText 459
ST_MPolyFromWKB 460
ST_MultiLineString 462
ST_MultiPoint 464
ST_MultiPolygon 465
ST_NumGeometries 467
ST_NumInteriorRing 468
ST_NumLineStrings 469
ST_NumPoints 470
ST_NumPolygons 471

空間処理関数 (続き)

ST_Overlaps 472
ST_Perimeter 474
ST PerpPoints 475
ST_Point 477
ST_PointFromText 480
ST_PointFromWKB 481
ST_PointN 482
ST_PointOnSurface 483
ST_PolyFromText 484
ST_PolyFromWKB 485
ST_Polygon 487
ST_PolygonN 489
ST_Relate 490
ST_RemovePoint 491
ST_SRID
ST_SrsId 493
ST_SrsID
ST_SRID 493
ST_SrsName 494
ST_StartPoint 495
ST_SymDifference 496
ST_ToGeomColl 498
ST_ToLineString 499
ST_ToMultiLine 500
ST_ToMultiPoint 501
ST_ToMultiPolygon 502
ST_ToPoint 503
ST_ToPolygon 504
ST_Touches 505
ST_Transform 507
ST_Union 508
ST_Within 510
ST_WKBToSQL 512
ST_WKTToSQL 513
ST_X 514
ST_Y 515
ST_Z 517

空間ストアード・プロシージャ

使用すべきでない 545
Geodetic Extender によってサポートされる 223

空間データ

インポート 89
エクスポート 89
クライアントからサーバーへのトランスフォーム 521
検索および分析
インターフェース 127
関数 127
索引の活用 128
ジオコーディング 95
使用 8
説明 3, 4
データ・タイプ 83
列 83

- 空間データ (続き)
 - ST_GEOMETRY_COLUMNS 298
 - 空間列
 - アクセスするビューの使用 126
 - 空間参照系への登録 87
 - 作成 85
 - ジオコーディング 95
 - 測地データを入れる 183
 - 形状データのインポート 91
 - 形状の距離情報 346
 - 形状の索引情報 347
 - 形状表現、データ・フォーマット 533
 - 形状ファイル
 - データのエクスポート 93
 - 経度、測地
 - 定義 169
 - 検査
 - Spatial Extender のインストール 41
 - 格子索引
 - 概要 104
 - 作成 111
 - チューニング 115
 - 格子索引のチューニング
 - 索引アドバイザーを使用する 116
 - 構成パラメーター
 - 空間アプリケーション
 - 値 51
 - チューニング 51
 - コマンド
 - db2se 131
 - コマンド行プロセッサ (CLP)
 - メッセージ 157
 - Spatial Extender コマンド 131
 - コントロール・センター
 - メッセージ 159
- [サ行]**
- 最小外接円 (MBC)
 - 空間処理関数の結果 218
 - 定義 185
 - ST_Geometry 属性 209
 - 最小外接長方形 (MBR)
 - 空間グリッド・インデックスでの使用
 - 104
 - 定義 13
 - 索引
 - 空間インデックス統計を分析する 118
 - 空間グリッドの作成 111
 - 空間グリッドのための CREATE INDEX ステートメント 114
 - 空間グリッド・インデックス 104
 - 格子サイズを決定する 116
 - 索引アドバイザー・コマンド 123
 - 索引 (続き)
 - 測地ポロノイを作成するための
 - CREATE INDEX ステートメント
 - 190
 - 測地ポロノイ・セル構造 187
 - 索引アドバイザー
 - 空間インデックス統計を分析する 118
 - 格子サイズを決定する 116
 - 使用する場合 106
 - 目的 104, 115
 - GET GEOMETRY コマンドが実行する
 - 123
 - 索引を分析する
 - 索引アドバイザーを使用する 118
 - 作成
 - 空間グリッド・インデックス 111
 - 測地ポロノイ・インデックス 189
 - 座標
 - 空間参照系 70
 - 空間参照系での変換 70
 - 最大値および最小値の検出 77
 - 入手 331
 - パフォーマンスを向上させるための変換 75
 - 座標系
 - 概要 61
 - 作成 69
 - サポートされている 535
 - 選択 69
 - ST_COORDINATE_SYSTEMS カタログ・ビュー 297
 - ST_SPATIAL_REFERENCE_SYSTEMS カタログ・ビュー 305
 - 座標参照系
 - 緯度と経度 167
 - 参照データ
 - ジオコーダー 44
 - DB2 Spatial Extender
 - 57
 - アクセスの設定 57
 - サンプル・データ
 - Spatial Extender 45
 - ジオコーダー
 - 概要 95
 - 参照データ 44
 - 登録 58
 - バッチ・モードで実行する 100
 - ST_GEOCODERS カタログ・ビュー
 - 301
 - ST_GEOCODER_PARAMETERS カタログ・ビュー
 - 299
 - ST_GEOCODING カタログ・ビュー
 - 301
 - ST_GEOCODING_PARAMETERS カタログ・ビュー
 - 303
 - ST_SIZINGS カタログ・ビュー
 - 304
 - ジオコーディング
 - 概要 95
 - セットアップ 97
 - バッチ・モード 100
 - ジオコーディングで使用される公式 75
 - 子午線 169
 - 座標系 535
 - システム要件
 - Geodetic Extender のための 175
 - 事前割り当てテキスト (WKT) 表現、データ・フォーマット 527
 - 事前割り当てバイナリー (WKB) 表現、データ・フォーマット 532
 - 自動ジオコーディング 95, 99
 - シナリオ
 - Spatial Extender のセットアップ 17
 - 集約関数
 - 空間列 356, 518
 - 照会
 - 空間、サブミットするインターフェース 127
 - 空間インデックスの活用 128
 - 実行する空間処理関数 127
 - 使用可能化
 - 空間操作 55, 56
 - スケール因数
 - 新しい空間参照系の計算 77
 - 概要 75
 - 図形のプロパティ
 - 概要 13
 - 空間処理関数 331
 - 境界情報 335
 - 空間参照系 337
 - 構成情報 337
 - 座標および指標に関する情報 331
 - 図形内の図形 333
 - データ・タイプ情報 331
 - ディメンションに関する情報 336
 - ストアド・プロシージャ
 - 空間アプリケーション用呼び出し 142
 - 問題 153
 - 呼び出し
 - 空間アプリケーション 141
 - GSE_export_sde 238
 - GSE_import_sde 240
 - ST_alter_coordsys 242
 - ST_alter_srs 244
 - ST_create_coordsys 248
 - ST_create_srs 250
 - ST_disable_autogeocoding 257
 - ST_disable_db 258
 - ST_drop_coordsys 260
 - ST_drop_srs 261
 - ST_enable_autogeocoding 262
 - ST_enable_db 264
 - ST_export_shape 266

索引

ストアード・プロシージャ (続き)
ST_import_shape 270
ST_register_geocoder 278
ST_register_spatial_column 283
ST_remove_geocoding_setup 284
ST_run_geocoding 286
ST_setup_geocoding 289
ST_unregister_geocoder 293
ST_unregister_spatial_column 294
正角図法 67
正距図法 67
正積図法 67
世界の人口密度を基準にした
ポロノイ・セル構造 186
赤道 169
赤道地帯
表すポリゴン 209
設定
ジオコーディング操作 97
自動ジオコーディング 99
セットアップ
DB2 Spatial Extender 25
線 11
線形ユニット
座標系 535
測地緯度 169
測地学 167
測地基準
座標系 535
説明 167
ST_SPATIAL_
REFERENCE_SYSTEMS 305
測地系 168
座標系定義での 232
測地基準 167, 168
測地経度 169
測地座標系 61
測地参照系 167
測地参照系 ID
ST_create_srs 250
測地参照系 (SRS)
説明 70
測地線
サンプル 209
定義 171
測地データ
説明 4
表に入れる 183
測地での機能
ST_Area 358
ST_Buffer 368
ST_Contains 373
ST_Difference 379
ST_Distance 383
ST_Generalize 400
ST_Intersection 416

測地での機能 (続き)
ST_Intersects 418
ST_Length 427
ST_Perimeter 474
ST_SymDifference 496
ST_Union 508
ST_Within 510
測地ポリゴン 172
測地ポロノイ・インデックス
活用 128
空間グリッド・インデックスとの比較
103
作成 189
代替ポロノイ構造を選択する 187
利用する関数 185
CREATE INDEX ステートメント 190
測地ライセンスの使用可能化 175
測地領域
説明 172
測定情報の取得 331
ソフトウェア要件
Spatial Extender 26

[夕行]

第 180 子午線
交差する最小外接円 218
またぐ図形 209
第 180 子午線、と交差する線 209
楕円面
Geodetic Extender 232
タスク
Spatial Extender のセットアップ 17
地球全体
表す 209
地図
製品に付属しているサンプル 45
地図投影法
座標系 535
地理エクステンツの定義 77
地理機能
説明 3
データによる表現 4
データのエクスポート
データ
形状ファイル 93
SDE 転送ファイル 94
データベース
空間アプリケーション用の構成 51
空間アプリケーション用のセットアッ
プ 51
空間操作を使用可能にする 56
概要 55
空間データの移行 47

データベース構成パラメーター
空間アプリケーション
チューニング 51
APPLHEAPSZ パラメーター 51
APP_CTL_HEAP_SZ パラメーター
51
LOGFILSZ パラメーター 51
LOGPRIMARY パラメーター 51
LOGSECOND パラメーター 51
データベース・マネージャー構成
パラメーター、空間アプリケーション
用チューニング 51
データ・タイプ情報、入手 331
データ・フォーマット
形状表記 533
事前割り当てテキスト (WKT) 表記
527
事前割り当てバイナリー (WKB) 表記
532
Geography Markup Language
(GML) 534
データ・マップ
Spatial Extender 45
デフォルトの空間参照系 71
度
緯度と経度 169
投影座標系 61, 67
登録
空間列 87
ジオコーダー 58
トラブルシューティング
関数 156
管理通知ログ 162
形状情報メッセージ 157
マイグレーション・メッセージ 157
Spatial Extender
サンプル・プログラム 43
ストアード・プロシージャ 153
トレース 160
メッセージ 151
runGseDemo の使用 43
トランスフォーム・グループ
概要 521

[ハ行]

ハードウェア要件
Spatial Extender 26
バッチ。ジオコーディング 95
パフォーマンス
座標データの変換 75
パフォーマンスを向上させるための乗数
座標の処理 75
半球
表すポリゴン 209

比較関数
 概要 318
 コンテナ・リレーションシップ 320
 図形どうしの交差 323, 330
 図形のエンベロープ 328
 同一の図形 328
 DE-9IM パターン・マトリックス・ストリング 330

ビュー
 DB2 Spatial Extender
 空間列のアクセス 126

表
 空間列 85
 形状データのインポート 91

複数折れ線、Spatial Extender 同種集合
 11

複数ポイント、Spatial Extender 同種集合
 11

複数ポリゴン、Spatial Extender 同種集合
 11

プログラミング考慮事項
 Spatial Extender サンプル・プログラム
 141

ヘッダー・ファイル、DB2 Spatial
 Extender を含む 141

変換
 座標系間の空間データ 347
 座標処理の改善 75

ポイント 11

方位図法 67

方向が正確な投影 67

ポリゴン
 図形タイプ 11
 測地領域を定義する 172

ポロノイ分割 186

ポロノイ・セル構造
 索引のための代替を選択する 187
 説明 186

本初子午線 169

[マ行]

マイグレーション
 Spatial Extender 47, 49

メッセージ
 移行情報 157
 関数 156
 形状情報 157
 コントロール・センター 159

Spatial Extender
 ストアード・プロシージャ 153
 の一部 151
 CLP 157

問題を分離するイベントのトレース 160

[ラ行]

ライセンス
 Geodetic Extender のための 175

リング
 説明 13
 測地領域を定義する 172

例
 Spatial Extender 143

列
 空間データ 85

ログ
 診断 162

[ワ行]

和集約関数 518

A

AIX
 インストール
 DB2 Spatial Extender 29

APPLHEAPSZ 構成パラメーター
 チューニング 51

APP_CTL_HEAP_SZ パラメーター、チューニング 51

ArcExplorer
 インターフェースとして使用 127

AsShape、使用すべきでない空間処理関数
 573

C

constructor 関数
 概要 310
 ジオグラフィー・マークアップ言語
 (GML) 表記 317
 事前割り当てテキスト表現 313
 事前割り当てバイナリー表現 315
 ESRI 形状表記 316

COORD_REF_SYS 空間カタログ・ビュー、使用すべきでない 569

CREATE INDEX ステートメント
 空間グリッド・インデックス 114
 測地ポロノイ・インデックス 190

D

DB2 Geodetic Extender
 サポートされる空間処理関数 218

db2se コマンド 131

db2trc コマンド 160

DEFAULT_SRS
 空間参照系 73

DE_HDN_SRS_1004
 空間参照系 73

distance
 測地線によって計測する距離 171
 ST_Distance 機能 383

G

GCSW_DEUTSCHE_HAUPTDRE
 IECKSNETZ
 座標系 73

GCS_NORTH_AMERICAN_1927
 座標系 73

GCS_NORTH_AMERICAN_1983
 座標系 73

GCS_WGS_1984
 座標系 73

Geodetic Extender
 サポートされる空間カタログ・ビュー
 223
 サポートされる空間ストアード・プロシージャ 223
 使用する場合 168
 セットアップ 175
 説明 167
 楕円面 232
 違い 209
 ST_Geometry 属性 209

Geographic Markup Language (GML)、データ・フォーマット 534

geometries
 概要 11
 空間データ 8
 クライアント/サーバー・データ転送
 521
 新規の生成
 ある図形の別の図形への変換 338
 概要 338
 既存の指標に基づいた 344
 修正フォーム 345
 新規空間構成 339
 複数から 1 つ 343

プロパティ
 概要 13
 空間処理関数の形状の特性も参照
 331

GeometryFromShape、使用すべきでない空間処理関数 573

GEOMETRY_COLUMNS、使用すべきでない空間カタログ・ビュー 569

GET GEOMETRY コマンド
 構文 123

gseidx コマンド
 空間インデックス統計を分析する 118
 格子サイズを決定する 116

索引

`gse_disable_autogc` ストアード・プロシージャ 257
`gse_disable_autogc`、使用すべきでない空間
ストアード・プロシージャ 545
`gse_disable_db` ストアード・プロシージャ
 258
`gse_disable_sref` ストアード・プロシージャ
 261
`gse_disable_sref`、使用すべきでない空間ス
トアード・プロシージャ 545
`gse_enable_autogc` ストアード・プロシ
ージャ 262
`gse_enable_autogc`、使用すべきでない空間
ストアード・プロシージャ 545
`gse_enable_db` ストアード・プロシージャ
 264
`gse_enable_db`、使用すべきでない空間ストア
ード・プロシージャ 545
`gse_enable_idx`、使用すべきでない空間ス
トアード・プロシージャ 545
`gse_enable_sref` ストアード・プロシージャ
 250
`gse_enable_sref`、使用すべきでない空間ス
トアード・プロシージャ 545
`GSE_export_sde` ストアード・プロシージャ
 238
`gse_export_shape` 266
`GSE_import_sde` ストアード・プロシージャ
 240
`gse_import_sde` ストアード・プロシージャ
 240
`gse_import_shape` ストアード・プロシージャ
 270
`gse_import_shape`、使用すべきでない空間
ストアード・プロシージャ 545
`gse_register_gc` ストアード・プロシージャ
 278
`gse_register_gc`、使用すべきでない空間ス
トアード・プロシージャ 545
`gse_register_layer` ストアード・プロシージャ
 283
`gse_register_layer`、使用すべきでない空間
ストアード・プロシージャ 545
`gse_run_gc` ストアード・プロシージャ
 286
`gse_run_gc`、使用すべきでない空間ストア
ード・プロシージャ 545
`gse_unregist_gc` ストアード・プロシージャ
 293
`gse_unregist_gc`、使用すべきでない空間ス
トアード・プロシージャ 545
`gse_unregist_layer` ストアード・プロシ
ージャ 294
`gse_unregist_layer`、使用すべきでない空間
ストアード・プロシージャ 545

H

`h` ヘッダー・ファイル 141
`HP-UX`
インストール
DB2 Spatial Extender 32

I

`Is3d`、使用すべきでない空間処理関数
 573
`IsMeasured`、使用すべきでない空間処理関
数 573

L

`LineFromShape`、使用すべきでない空間処
理関数 573
`LocateAlong`、使用すべきでない空間処理
関数 573
`LocateBetween`、使用すべきでない空間処
理関数 573
`LOGFILSIZ` 構成パラメーター 51
`LOGPRIMARY` 構成パラメーター 51
`LOGSECOND` 構成パラメーター
チューニング 51

M

`migrate_v82` コマンド
説明 49
`MLineFromShape`、使用すべきでない空間
処理関数 573
`MPointFromShape`、使用すべきでない空間
処理関数 573
`MPolyFromShape`、使用すべきでない空間
処理関数 573
`M`、使用すべきでない空間処理関数 573

N

`NAD27_SRS_1002`
空間参照系 73
`NAD83_SRS_1`
空間参照系 73

P

`PointFromShape`、使用すべきでない空間処
理関数 573

S

`SDE` 転送ファイル
からデータのインポート 92
データのエクスポート 94
`ShapeToSQL`、使用すべきでない空間処理
関数 573
`Solaris` オペレーティング環境
インストール
DB2 Spatial Extender 34
`Spatial Extender`
インストール 37
参照データ 57
アクセスの設定 57
使用可能化 56
使用する場合 168
で提供されている空間参照系 73
`SPATIAL_GEOCODER`、使用すべきでない
空間カタログ・ビュー 569
`SPATIAL_REF_SYS`、使用すべきでない空
間カタログ・ビュー 569
`SQL` ステートメント
測地ボロノイ・インデックスを利用す
る 190
`ST_alter_coordsys` ストアード・プロシージャ
 242
`ST_alter_srs` 244
`ST_COORDINATE_SYSTEMS` 297
`ST_create_coordsys` ストアード・プロシ
ージャ 248
`ST_create_srs` 250
`ST_disable_autogeocoding` 257
`ST_disable_db` ストアード・プロシージャ
 258
`ST_Distance` 383
`ST_drop_coordsys` ストアード・プロシ
ージャ 260
`ST_drop_srs` 261
`ST_enable_autogeocoding` ストアード・プ
ロシージャ 262
`ST_enable_db` ストアード・プロシージャ
 264
`ST_export_shape` ストアード・プロシージャ
 266
`ST_GEOCODERS` 301
`ST_GEOCODER_PARAMETERS` 299
`ST_GEOCODING` 301
`ST_GEOCODING_PARAMETERS` 303
`ST_Geometry` 属性
測地での相違 209
`ST_GEOMETRY_COLUMNS` 298
`ST_GeomFromText`、使用すべきでない空
間処理関数 573
`ST_GeomFromWKB`、使用すべきでない空
間処理関数 573

ST_import_shape ストアード・プロシージャ 270

ST_LineFromText、使用すべきでない空間処理関数 573

ST_LineFromWKB、使用すべきでない空間処理関数 573

ST_MLineFromText、使用すべきでない空間処理関数 573

ST_MLineFromWKB、使用すべきでない空間処理関数 573

ST_MPointFromText、使用すべきでない空間処理関数 573

ST_MPointFromWKB、使用すべきでない空間処理関数 573

ST_MPolyFromText、使用すべきでない空間処理関数 573

ST_MPolyFromWKB、使用すべきでない空間処理関数 573

ST_OrderingEquals、使用すべきでない空間処理関数 573

ST_PointFromText、使用すべきでない空間処理関数 573

ST_Point、使用すべきでない空間処理関数 573

ST_PolyFromText、使用すべきでない空間処理関数 573

ST_PolyFromWKB、使用すべきでない空間処理関数 573

ST_register_geocoder ストアード・プロシージャ 278

ST_register_spatial_column ストアード・プロシージャ 283

ST_remove_geocoding_setup ストアード・プロシージャ 284

ST_run_geocoding ストアード・プロシージャ 286

ST_setup_geocoding ストアード・プロシージャ 289

ST_SIZINGS 304

ST_SPATIAL_REFERENCE_SYSTEMS 305

ST_SymmetricDiff、使用すべきでない空間処理関数 573

ST_Transform、使用すべきでない空間処理関数 573

ST_UNITS_OF_MEASURE 307

ST_UNITS_OF_MEASURE カタログ・ビュー 307

ST_unregister_geocoder ストアード・プロシージャ 293

ST_unregister_spatial_column ストアード・プロシージャ 294

W

WGS84_SRS_1003
空間参照系 73

Windows
インストール
DB2 Spatial Extender 28

Z

Z、使用すべきでない空間処理関数 573

IBM と連絡をとる

技術上の問題がある場合は、お客様サポートにご連絡ください。

製品情報

DB2 Universal Database 製品に関する情報は、
<http://www.ibm.com/software/data/db2/udb> から入手できます。

このサイトには、技術ライブラリー、資料の注文方法、製品のダウンロード、ニュースグループ、フィックスパック、ニュース、および Web リソースへのリンクに関する最新情報が掲載されています。

米国以外の国で IBM に連絡する方法については、IBM Worldwide ページ (www.ibm.com/planetwide) にアクセスしてください。



Printed in Japan

SC88-9171-01



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